遺伝的アルゴリズムによる
農作業計画の最適化に関する研究

2002

大土井克明

遺伝的アルゴリズムによる
農作業計画の最適化に関する研究

2002

大土井克明

目次

第1章　緒言

1.1　本研究の背景

農業は，自然の生態系を利用して人間生活に不可欠な食料を生産する産業であるため，地域の自然条件，特に気温，降水量などの気候条件や地形などの国土条件に即してその形態が形成され発展する [1]．国土の大部分がアジアモンスーン地帯に属する我が国においては，高温多雨な夏期の気候条件が熱帯地方原産の稲の栽培を可能とし，品種改良により全土で稲作を中心とする水田農業が発展してきた．我が国の国土はその61%が山地であり，土地利用のしやすい平地が限られるため，農地と他産業や住宅地などの土地利用との競合が激しい．そのため，平地だけでなく中山間傾斜地まで農業用地として利用されているが，国土全体に対する農地の構成比は約13% [2] と欧米諸国と比較して著しく少なく，また微減傾向にある．また，戦後行なわれた農地改革により北海道では4町歩，内地では1町歩を超える農地の所有を禁じられたため，北海道を除く我が国の水稲の経営耕地規模別の農家数は1ha未満のものが約77%を占めており [3]，小規模な経営形態が我が国の農業の特徴の一つである．

第二次世界大戦終結後，朝鮮戦争特需により我が国の工業は飛躍的な成長を始め，農村から都市部へ労働力が大量に流出したため，農業における労働力不足が問題となった．これを補うため農業の機械化が促進され，昭和40年代前半には耕うん，田植え，管理，収穫，乾燥，調整の稲作機械化一貫体系が完成し農業の省力化が進んだ．省力化により余剰となった労働力は第二次産業，第三次産業に進出し，我が国の高度経済成長を支えたが，この経済成長により農村労働者と都市労働者の間に大きな賃金格差を生じることとなり，余剰の労働力のみならず後継者まで賃金の高い他産業に主たる収入を求め，農業の担い手不足が深刻な問題となった．一方，この賃金格差を解消するために，生産者米価が昭和48年に約15%，49年には約32%と大幅に引き上げられ農家の収入を補ったが，そのために我が国の米価は大きく上昇し，また，経済成長により円高が急速に進んだため国際価格との間に大きな格差を生じることとなった．このような中で，平成5年にはウルグアイラ

ウンド合意により主食の米についてもミニマムアクセスが義務づけられ，安価な外国産の米との競争を余儀なくされることとなった．

このように現在の我が国の農業は，後継者不足による農業従事者の減少・高齢化，および外国産農産物に対する競争力不足という深刻な問題を抱えており，危機的な状況にあると言える．しかし，農業は食料を生産する産業であるとともに，国土の保全，水源のかん養，自然環境の保全といった多面的な機能を有しており，安定した生活のために重要な役割を果たしている．したがって，これらの問題点を克服し，農業の持続的発展を図る必要がある．そのため，農業従事者の減少・高齢化への対策として農業の省力化が，外国産農産物に対する競争力不足への対策として農業の低コスト化が強く求められている．

農業の省力化を実現するために有効な手段として，機械化されていない作業の機械化の推進，農業機械の高性能化が挙げられる．稲作の作業は大部分が機械化されているが，効率的な作業を行なうために畦越えのできる走行装置の開発 [4,5] や，無人での作業を可能とする田植機の開発 [6–10] など農業機械の高性能化を目的とした研究が盛んに行なわれている．しかし，機械の高性能化にともないその価格も上昇する．前述したように経営規模が小さい我が国の農業の現状では，これらの高性能な機械は生産コストを上昇させる原因となり，農家が単独で導入するのは困難である．このような省力化と低コスト化という相反する目的を達成するために，機械の共同利用や共同作業を行なう集落営農組織や，農作業の受託を行なう農業サービス事業体などにより経営規模を拡大し，機械の稼働率を上げることにより生産コストを削減する取り組みが各地で行なわれている．これらの組織においては，作業効率の高い機械を導入するとともに，導入した機械の効率的な運用を可能にする機械化一貫作業体系の確立が必要である．つまり，経営規模が大きくなるにつれて，特に作業受託組織においては，作業体系は複雑化するため，数多くの圃場を対象として機械群をいかに適切に運用できるかが重要となる．しかし，食料・農業・農村白書によると我が国においては耕地面積，農家人口，農業粗生産額のいずれについても，農業の約4割が平野の外縁部から山間地に至るいわゆる中山間地で行なわれている．これらの地域では地形的に大区画化のできない圃場が数多く存在するため，作業受託などにより経営規模を拡大すると効率の良い作業計画を立てるのが困難となる．

本研究は，作業受託などにより経営規模を拡大して省力化・低コスト化を図る際に課題となる，効率的な農業機械の運用を可能とする作業計画の構築を最適化アルゴリズムにより実現するための考察を行ない，アプリケーションの開発を行なったものである．対象と

した営農組織は中山間地域に所在する農作業受託組織で，この組織が作業を受託した圃場の位置，形状，および面積，各作業での圃場作業量などの基礎的データを収集した．これらのデータを用いて開発した手法の性能を検討した．

1.2 遺伝的アルゴリズムの概要と応用例

本研究では最適化の手法として，組合せ最適化問題に適した遺伝的アルゴリズム (Genetic Algorithm:GA) を用いた．GA は生物の進化の過程に見られるいくつかの現象を模倣した手法で，最適化の対象となる問題は，遺伝子の並びである染色体にコード化され，各遺伝子が 0 か 1 の 2 値をとるビット列，ある範囲の整数値や記号となる文字列の形で表現される．図 1.1 に示したものは各遺伝子が 0 か 1 のビット値をとり，遺伝子長が 10 の染色体である．なお，ある個体の染色体に特有な遺伝子の構成と配列をその個体の遺伝子型 (genotype) といい，環境 (最適化の対象となる問題) の中で遺伝子情報に基づいて発現される形質をその個体の表現型 (phenotype) という．このような染色体を持つ個体 N_p 個からなる集団を考え，この集団に対して交叉 (crossover)，突然変異 (mutation) などの遺伝的操作を行ない，適応度に応じて生き残る個体を選択して次世代の集団とする．

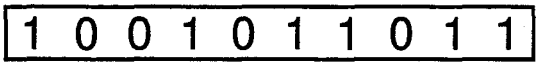


図 1.1: コード化
Figure 1.1: Coding

図 1.2 は交叉の例で最も単純な一点交叉を示している．図 1.2 の Pa, Pb は親となる個体の染色体で，ランダムに設定される交叉点以降の遺伝子を入れ換えることで新たに二つの

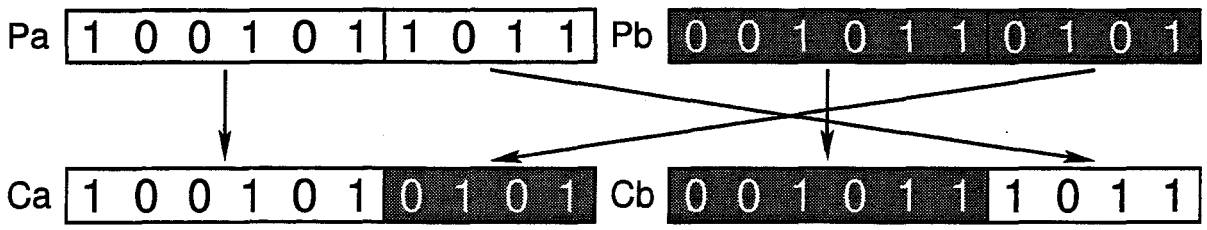


図 1.2: 一点交叉
Figure 1.2: One-point crossover

子となる個体の染色体Ca, Cbを作る．この例では交叉点は6番目と7番目の遺伝子座の間に設定されており，CaはPaの前半部とPbの後半部，CbはPbの前半部とPaの後半部を受け継いでいる．交叉の方法にはこの他に図1.3の二点交叉をはじめとする多点交叉，図1.4のような設定したマスクの値によりどちらの親の遺伝子を継承するかを決定する一様交叉 [11] などがある．図1.4の例ではマスクとして0か1の値となるビット列を設定し，Caではマスクが0の遺伝子座ではPaの，1の遺伝子座ではPbの遺伝子を継承し，Cbはその逆の遺伝子を継承している．

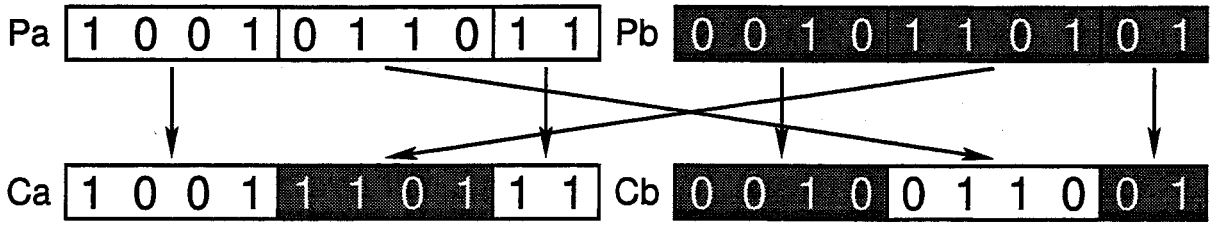


図 1.3: 二点交叉
Figure 1.3: Two-point crossover

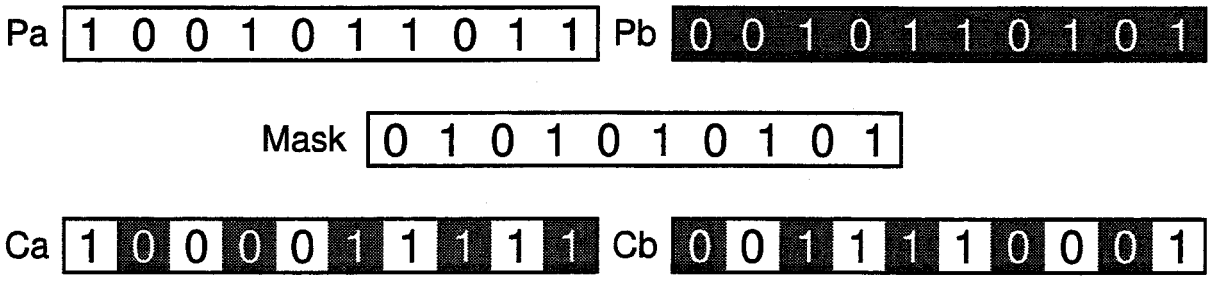


図 1.4: 一様交叉
Figure 1.4: Uniform crossover

突然変異はある確率で選択された遺伝子座の遺伝子を対立遺伝子に変化させる操作が一般的で，図1.5の例では選択された遺伝子座の遺伝子が0であるため対立遺伝子である1に変化させている．この他に，個体の遺伝子の並びに意味があり，単独の遺伝子を変化させることができないような場合には，図1.6のように二つの遺伝子座の間の遺伝子の並びを反転させる方法などを用いる．この一連の操作 (交叉，突然変異，選択) を多数回繰り返すことで適応度の高い個体が現れ，解を得ることができる．

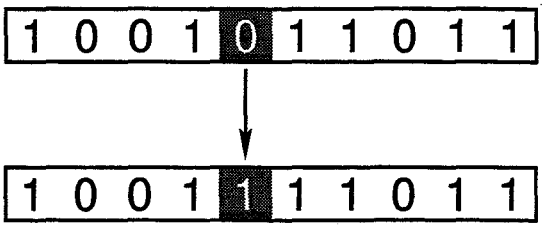


図 1.5: 突然変異
Figure 1.5: Mutation

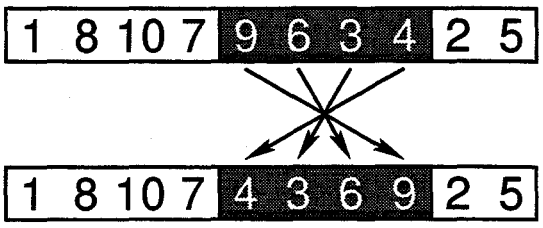


図 1.6: 突然変異 (逆位)
Figure 1.6: Mutation (Inversion)

GA の研究は 1960 年代の Holland による適応システムの研究 [12] に始まり，その後 1989 年に Goldberg [13] によって枠組が整理された．GA の応用は最適化問題が主流であり，巡回セールスマン問題やナップザック問題といった NP(Nondeterministic Polynomial) 完全問題に適用した研究が多く見られる．また，産業的応用も試みられており，ジョブショップスケジューリング問題 [14,15] や LSI の素子配置問題 [16] など様々な分野に適用されている．農業機械の分野では，農用車両を自律走行させる時に必要となる作業経路の事前スケジューリングについて，積算舵角変化量を最小化させる最適作業経路を GA により求める研究 [17] が行なわれている．

1.3 作業計画に関する研究の沿革

前述したとおり，我が国の農業が抱える問題点を克服するためには，省力化・低コスト化という要求を満たす必要があり，農業機械の高性能化とともに効率的な運用による機械コストの削減が不可欠となる．これを実現するために，作業体系シミュレータ [18] を用いて様々な経営形態でシミュレーションし，生産コストの分析や収益性の分析，また，機

械の更新および導入計画などに活用する研究 [19–21] が行なわれている．石束は幾種類かの作物，あるいは品種が作付けられる比較的規模の大きな生産現場を想定し，そのような現場で最も重要な課題となる作付作物の組み合わせや作物生産の安定化を行なうために，所有する農業機械や施設の規模および労動力が適正であるかを判断するシミュレータを開発した．このシミュレータでは日を単位にシミュレーションを進行させ，ある日に実施すべき作業を選びだし，優先度の高い作業から順に天候や土壌条件などを考慮して作業が可能か判断して作業ができる場合には機械，労力，資材の資料量を求めるもので，この処理を1年にわたって繰り返すものである．張らはこのシミュレータを大規模畑作農家，集団営農，専業農家からなる営農組合などの経営形態に対して適用し，経営改善について検討した．

樋口らは労働生産性の向上を目的とし，Hopfield型 (相互結合型) ニューラルネットワークを応用して最短巡回経路を求めた [22,23]．これは圃場間の移動距離を短くすることで移動に必要な時間を短縮し，実作業率を高くすることで労働生産性を向上させるというものであった．

1.4 目的と構成

従来，生産現場では前節で述べたような意思決定事項を経験に頼る方法で処理してきたが，経営規模の拡大とともに様々な拘束条件などにより問題が複雑になるため，合理的な判断を行なう意思決定支援システムが必要となる．本研究は，作業受託などにより経営規模を拡大して省力化・低コスト化を図る際に課題となる，効率的な農業機械の運用を可能とする作業計画を決定する意思決定支援システムの構築を目的としている．この作業計画の最適化を行うために，組み合わせ最適化問題に適したGAを応用した．

まず，第2章では農業機械が割り当てられた圃場を巡回し作業をする場合の最短経路を求める問題を，コード化，交叉の方法の異なる二つの手法により求め，各手法の特性，およびパラメータを変化させた時の性能の違いを考察し，適切な手法とパラメータを決定した．

第3章では，複数の機械で作業する場合の各機械への作業を行なう圃場の割り当てについて言及する．農作業計画を最適化する場合，作業適期，天候，水管理など様々な条件を考慮する必要があるが，ここでは単純化して一日の実作業時間と割り当てられた圃場での

作業にかかる時間との誤差を最小にすること，圃場間の移動距離を最短にすることの二つの条件に焦点を当て最適化を行なった．これは少なくともこれらの二つを考慮しないと，実行可能な解が得られないためである．二種類の手法を考案し，それらの性能を比較するとともに，計算時間短縮のために評価関数の単純化を行なった．

第4章では，第3章で行なった評価関数の単純化についての考察を行なった．個体の適応度を計算する評価関数はGAの性能に大きな影響を与えるため，適切な評価関数を用いる必要がある．したがって，第3章で用いた評価関数が適切に個体を評価していることを確認するために，単純化した計算方法で得られる評価と本来用いるべき値とを比較してその有効性を確認した．また，その計算方法が特定の圃場配置，圃場数，面積に関わらず有効であることを確認するとともに，第3章の手法に汎用性があることを確認するために，圃場をランダムに50箇所，および100箇所配置した仮想の圃場データを用いて計算し検証した．

参考文献

[1] 農林統計協会：図説 食糧・農業・農村白書(平成11年度版), 財団法人農林統計協会, (2000)

[2] 国土庁：土地白書(平成12年版), 国土庁, (2000)

[3] 農林水産省統計情報部：第74次農林水産省統計表平成9年～10年, 財団法人農林統計協会, (1999)

[4] 山崎稔, 笈田昭, 清水浩, 藤本寛央, 福嶋恵一：電気駆動不整地走行車両の開発研究 ―畦乗り越え時の制御について―, 農業機械学会関西支部報, 84, pp.1–2, (1998)

[5] 山崎稔, 笈田昭, 清水浩, 宮坂寿郎, 伊吹拓：電気駆動不整地走行車両の水平制御に関する研究, 農業機械学会関西支部報, 86, pp.3–4, (1999)

[6] 長坂善禎, 大谷隆二, 重田一人, 谷脇憲：ハイブリッドセンシングによる水田作業車の自律走行の研究(第1報), 第56回農業機械学会年次大会講演要旨, pp.321–322, (1997)

[7] 長坂善禎, 大谷隆二, 重田一人, 谷脇憲：ハイブリッドセンシングによる水田作業車の自律走行の研究(第2報), 第57回農業機械学会年次大会講演要旨, pp.405–406, (1998)

[8] 長坂善禎, 谷脇憲, 大谷隆二, 重田一人, 久保昭博, 中川渉, 小山智弘：ハイブリッドセンシングによる水田作業車の自律走行の研究(第3報), 第58回農業機械学会年次大会講演要旨, pp.117–118, (1999)

[9] 長坂善禎, 谷脇憲, 大谷隆二, 重田一人, 佐々木泰弘：自動走行田植機の開発(第1報) ―リアルタイムキネマティックGPSによる位置認識と自動走行―, 農業機械学会誌, 61(6), pp.179–186, (1999)

[10] 長坂善禎, 谷脇憲, 大谷隆二, 重田一人：自動走行田植機による移植作業, 第59回農業機械学会年次大会講演要旨, pp.345–346, (2000)

[11] Syswerda, G.：Uniform crossover in Genetic Algorithms, Proc. of 3rd Int. Conf. on Genetic Algorithms, pp.2–9, (1989)

[12] J. H. Holland : Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, (1975)

[13] D. E. Goldberg : Genetic Algorithms in Search, Optimization and Learning, Addison-Wesley, (1989)

[14] 西川禕一, 玉置久 : ジョブショップ・スケジューリング問題に対する遺伝アルゴリズムの一構成法, 計測自動制御学会論文集, 27(5), pp.593–599, (1991)

[15] 西川禕一, 玉置久 : 近傍モデルによる遺伝アルゴリズムの並列化とジョブショップ・スケジューリング問題への応用, 計測自動制御学会論文集, 29(5), pp.589–595, (1993)

[16] K. Handa and S. Kuga : Polycell placement for analog LSI chip designs by genetic algorithms and tabu search, Proc. of IEEE Int. Conf. on Evolutionary Computation, pp.716–721, (1995)

[17] 野口伸, 石井一暢, 寺尾日出男 : ニューラルネットワークによる農用車両の最適制御 (第 2 報) —遺伝的アルゴリズムを用いた作業経路計画—, 農業機械学会誌, 56(2), pp.83–92, (1994)

[18] 石束宣明 : 作業体系シミュレータ, 農業機械学会誌, 48(1), pp.107–113, (1986)

[19] 張鉄中, 米村純一, 笹尾彰, 石束宣明 : 農業機械の運営管理と利用方式の合理化に関する研究 (第 1 報) —シミュレーションによる大規模畑作農家の営農改善—, 農作業研究, 26(3), pp.190–195, (1991)

[20] 張鉄中, 笹尾彰, 石束宣明, 米村純一 : 農業機械の運営管理と利用方式の合理化に関する研究 (第 2 報) —シミュレーションによる集団営農の経営改善—, 農作業研究, 27(1), pp.65–74, (1992)

[21] 張鉄中, 笹尾彰, 米村純一 : 農業機械の運営管理と利用方式の合理化に関する研究 (第 3 報) —専業農家からなる営農組合の規模拡大と経営改善—, 農作業研究, 27(3), pp.218–226, (1992)

[22] 山崎稔, 笈田昭, 樋口昌吾 : ニューラルネットワーク手法によるトラクタの圃場巡回問題へのアプローチ, 農業機械学会関西支部報, 74, pp.99–100, (1993)

[23] 樋口昌吾：ニューラルネットワーク手法によるトラクタの圃場巡回問題へのアプローチ, 京都大学農学部卒業論文, (1993)

第2章　点在する圃場の最短巡回経路

2.1　はじめに

企業が工業製品を製造する場合，市場での需要を調査し，必要な部品や原材料および工場の製造ラインを確保して収益が最大となるように製造計画を立てる．製造工程では各工程の依存関係などを調査し，効率良く製造するために作業計画を最適化するのが一般的である．しかし農業では対象となる製品が農産物という生物であり，製造ラインにあたる圃場は気候などの自然条件の影響を受けるため，そのような最適化はほとんど行なわれていないのが現状である．

作業受託などにより経営規模を拡大し低コスト化を図るためには，農業機械に対する過剰投資を避け，所有機械の稼働率を高める必要がある．農業における各作業には作業適期があるため，この期間内に終了できる作業量には限界があるが，早稲，中稲，晩稲の品種を組み合わせて栽培することで作業適期を長くし，作業量を増やすことができる．したがって，農業での作業計画を考える場合，1.圃場面積，水資源，作業適期を考慮し各圃場に作付する品種を決定することが重要となる．また，経営規模の拡大により作業する圃場の数，所有する機械の台数が増えることから，どの圃場をいつどの機械で作業をするかの組合せは膨大な数となる．よって，2.所有する機械の効率的な運用をするために，自然条件や圃場の立地条件などの様々な拘束条件を考慮した上で各圃場を作業する機械の決定が必要となる．また，第1章で述べたように我が国の農業は約4割が中山間地で行なわれており，圃場が点在している場合が多い．特に，作業受託を行なう場合，委託される圃場が点在しがちであり，圃場間の移動にかかる時間が多くなる．農業機械の移動速度は低速であるため，運搬用のトラックなどに積載する時間などを考慮しても自走による移動にかかる時間より短時間で済む時は，運搬車に積載して移動するのが一般的である．したがって，この圃場間の移動距離を短くすることで自走して移動する部分を増やし，結果として一日の実作業時間を増やすことができる．よって，3.機械が割り当てられた圃場を順に作業する時の最適な作業順序の決定が必要となる．

本章では，これらのうち3.の機械が割り当てられた圃場を順に作業する時の最適な作業順序の決定について述べる．一日の作業は，機械の保管庫を出発し，作業する圃場を順に巡回して保管庫に戻る．この間，一度作業した圃場に戻ることはないため，この一連の移動経路を最短にする問題は，巡回セールスマン問題 (Travelling Salesperson Problem:TSP) として解くことができる．また，作業に二日以上かかる場合，一日ごとに保管庫に戻るが，広範囲での作業を受託している場合などでは，保管庫から作業する集落までの往復は運搬車で運ぶため，この往復にかかる時間を無視して作業する圃場間の移動だけを考えることでTSPに近似の問題として扱うことができる．TSPとはセールスマンが複数の都市を訪れる時の最短経路を求める問題で，その拘束条件は，1.全ての都市を訪れ，2.一度訪れた都市は二度訪れない，の二点である．この問題の解の候補となる巡回経路の組合せは，都市の数がN_fの時$(N_f-1)!/2$通り存在するため，N_fが大きくなると最短経路を得るのが困難となる．山崎ら [1] はトラクタの圃場巡回問題を解くために，Hopfield型(相互結合型)ニューラルネットワーク手法を応用した．Hopfield型ニューラルネットワークは階層型ニューラルネットワークとは異なり，入力層，隠れ層，出力層といった階層はなく，全てのユニットが相互に結合して値の受渡しを行なう手法である．しかし，この手法では入力するパラメータが多く，また，個々のパラメータを決定するのに工夫が必要であった．また，作業をする圃場の数が変化するとパラメータを変更する必要があり，圃場の数が多くなるとTSPの条件である一ヶ所一訪問を満足させることが困難になるという問題点があった．そこで，本研究ではこの問題点を解消するため，コード化や交叉の方法を工夫することで少なくとも実行可能な解を得ることができるGAを用いた．

2.2 GA適用における手法

GAを用いてTSPを解く時に問題となるのは実行不可能な経路を表わす個体，すなわち全ての都市を一度だけ訪れるという拘束条件を満たさない個体の生成である．このような経路を表わす個体の染色体は致死遺伝子を持つという．TSPを解く時のコード化の方法として，遺伝子座が作業をする順番に対応し遺伝子を圃場番号とする経路表現が遺伝子型と表現型が一致するため最も単純である．このコード化で最も単純な一点交叉を用いると，交叉により生成される個体の染色体は致死遺伝子を持つものがほとんどとなる．この致死遺伝子を抑制するためにはコード化あるいは交叉の方法に何らかの工夫が必要とな

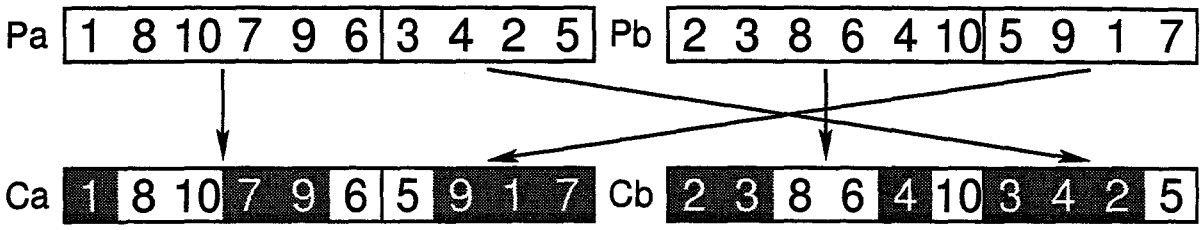


図 2.1: 致死遺伝子を持つ染色体
Figure 2.1: A choromosome having lethal genes

る．図2.1でPa, Pbは親となる個体が持つ染色体で，Paは圃場番号1, 8, 10, 7, 9, 6, 3, 4, 2, 5の順に作業をすることを表している．これらの二つの染色体を交叉させて子となる個体の染色体Ca, Cbを生成するが，Caが示す経路では1, 7, 9の圃場を二回訪れ，2, 3, 4の圃場には訪れていないため拘束条件を満たしておらず，致死遺伝子を持つ染色体となる．Cbも同様である．この致死遺伝子の発生を抑制するために，二点交叉を基本にして順列としての正当性を維持する手続きを加えたPartially Mapped Crossover (PMX) [2]，正当な順列を維持する手続きにおいて遺伝子の出現順序をできるだけ保存するOrder Crossover (OX) [3]，任意の置換が同じ文字を含まないいくつかの巡回置換の積に分解できる性質を利用して，一つの巡回置換に含まれる遺伝子を相互に交換するCycle Crossover (CX) [4]などの交叉演算子を工夫した方法や，都市リストを用いてコード化を工夫した方法 [5]などが報告されている．本研究ではこれら交叉演算子を工夫した方法およびコード化を工夫した方法からそれぞれ一つを選び，1. コード化を経路表現としOXを拡張したもの (手法1)，2. コード化を順序表現とし一点交叉を用いたもの (手法2)，の二つの手法で計算し，解を比較した．

2.2.1 コード化を経路表現とした手法 (手法1)

コード化

この手法ではコード化を図2.2に示す経路表現とし，致死遺伝子の生成を抑制するためにOXを拡張した交叉演算子を用いた．図2.2の染色体は遺伝子長が10，すなわち圃場数が10の時のもので，各遺伝子座は巡回順序，各遺伝子は圃場番号を示す．この染色体を持つ個体は遺伝子型と表現型が一致し，圃場番号1, 8, 10, 7, 9, 6, 3, 4, 2, 5の順に作業をすることを表している．

1	8	10	7	9	6	3	4	2	5

図 2.2: 手法 1 の染色体
Figure 2.2: A chromosome of the method 1

交叉

OX は基本的に二点交叉で，順列としての正当性を維持するために手続きにおいて遺伝子の出現順序をできるだけ保存しようとするものである．OX では以下の手順で交叉を行なう．

手順 1 i番目とj番目 $(i < j)$ の二つの遺伝子座をランダムに選んでそれぞれの直後を交叉点とする．

手順 2 $k = i + 1$，$l = i + 1$ に初期化する．

手順 3 $g_{Ca}(k) = g_{Pa}(l)$，$k = k + 1$，$l = l + 1$ とする．$k \leq j$ なら手順 3 へ，そうでなければ手順 4 へ進む．

手順 4 $g_{Pb}(l)$ について，$g_{Pb}(l) = g_{Pa}(m)$ となる m が $i + 1 \leq m \leq j$ であれば，$l = l + 1$ として手順 4 へ，そうでなければ手順 5 へ進む．

手順 5 $g_{Ca}(k) = g_{Pb}(l)$ とする．$k = i$ であれば終了，そうでなければ $k = k + 1$，$l = l + 1$ として手順 4 へ進む．

手順 6 Ca を得る．$g_{Ca}(x), g_{Pa}(x), g_{Pb}(x)$ をそれぞれ $g_{Cb}(x), g_{Pb}(x), g_{Pa}(x)$ に置き換えて手順 2〜手順 5 を繰り返し，Cb を得る．

ここで

Pa, Pb, Ca, Cb : 親 a，親 b，子 a，子 b の染色体

$g_{Pa}(x), g_{Pb}(x), g_{Ca}(x), g_{Cb}(x)$: Pa, Pb, Ca, Cb の x 番目の遺伝子の値

L : 遺伝子長 (圃場数)

k : 子となる染色体の遺伝子座を指すカウンタ $(1 \leq k \leq L, k > L$ の時 $k = k - L)$

l : 親となる染色体の遺伝子座を指すカウンタ $(1 \leq l \leq L, l > L$ の時 $l = l - L)$

である．

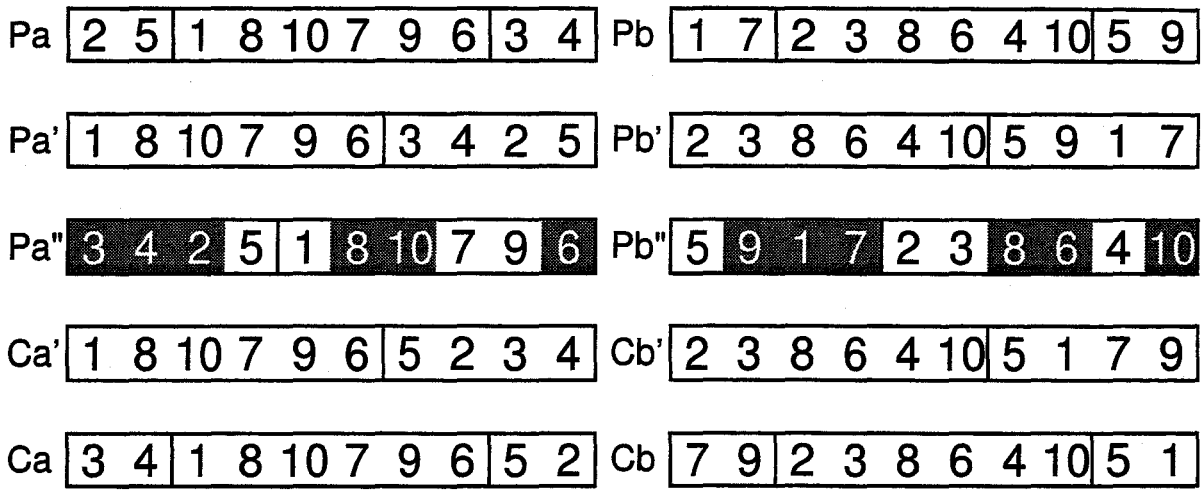


図 2.3: Order Crossover
Figure 2.3: Order Crossover

この交叉演算子の手順を図2.3に示す．この図のPa, Pbが親となる個体の染色体で2番目と8番目の遺伝子座の直後に交叉点がある．図のPa', Pb'はPa, Pbの変形であるが，TSPの解となる経路は閉路であるためPa, Pbと同じ経路を表すものである．よってPa, Pbでの二点交叉は，Pa', Pb'で6番目の遺伝子座の直後に交叉点のある一点交叉と等価である．単純な一点交叉の場合，図2.1の様に交叉点以降の遺伝子を入れ換えるため致死遺伝子を生成する危険性が高いが，この演算子では，交叉点直後の遺伝子を先頭にしたPa", Pb"を作り，Pa", Pb"からそれぞれPb', Pa'の前半部にある遺伝子を除くことにより致死遺伝子の生成を抑制する．子となる個体と同じ経路を示す染色体Ca'はPa'の前半部とPb"からPa'の前半部にある遺伝子を取り除いたものから，Cb'はPb'の前半部とPa"からPb'の前半部にある遺伝子を取り除いたものから生成する．Ca, CbはCa', Cb'の先頭の遺伝子がPa, Pbの一つ目の交叉点直後の遺伝子座にくるように変形して得られる．

このようにOXは交叉点以降で致死遺伝子の生成を抑制しながら，親となる個体の染色体での遺伝子の出現順序を保存する交叉演算子である．図2.3でCa'の交叉点以降の遺伝子は5, 2, 3, 4であるが，2, 3, 4, 5でもこの二つの条件を満たす交叉となる．このような交叉は，交叉点以降の遺伝子座がN個あればN通り存在するが，OXではCa'の後半部分はPb'の交叉点直後の遺伝子座を起点としているため5, 2, 3, 4となっている．しかしそのため交叉点の直前直後の二つの遺伝子の並びには親の遺伝子の出現順序が反映されていない．Ca'で交叉点の直前の遺伝子は6で5, 2, 3, 4と続いているが，Pbでの遺伝子6の後の

出現順序は4, 5, 2, 3であり，Ca'にこれを反映させることでより多く親の形質を継承することができる．そこで本研究ではこの二つの遺伝子の並びが親の形質を継承するように変更した [6]．本研究で用いた交叉は以下の手順で行なった．

手順1 i番目の遺伝子座をランダムに選んでその直後を交叉点とする．

手順2 $k=1$，$l=1$に初期化する．

手順3 $g_{Ca}(k)=g_{Pa}(l)$，$k=k+1$，$l=l+1$とする．$k \le i$なら手順3へ，そうでなければ手順4へ進む．

手順4 $g_{Pa}(i)=g_{Pb}(m)$となるmを探し，$l=m+1$とする．

手順5 $g_{Pb}(l)$について，$g_{Pb}(l)=g_{Pa}(n)$となるnが$n \le i$であれば，$l=l+1$として手順5へ，そうでなければ手順6へ進む．

手順6 $g_{Ca}(k)=g_{Pb}(l)$とする．$k=L$であれば終了，そうでなければ$k=k+1$，$l=l+1$として手順5へ進む．

手順7 Caを得る．$g_{Ca}(x), g_{Pa}(x), g_{Pb}(x)$をそれぞれ$g_{Cb}(x), g_{Pb}(x), g_{Pa}(x)$に置き換えて手順2〜手順6を繰り返し，Cbを得る．

この交叉演算子の手順を図2.4に示す．Pa, Pbが親となる個体の染色体であり，6番目の遺伝子座の直後に交叉点が設定されている．TSPでの二点交叉は一点交叉と等価であるため本研究では一点交叉を基本とした．OXとの相違点は，OXでは交叉点の直後の遺伝子座を起点として交叉点以降の遺伝子の出現順序を決定しているのに対して，この演

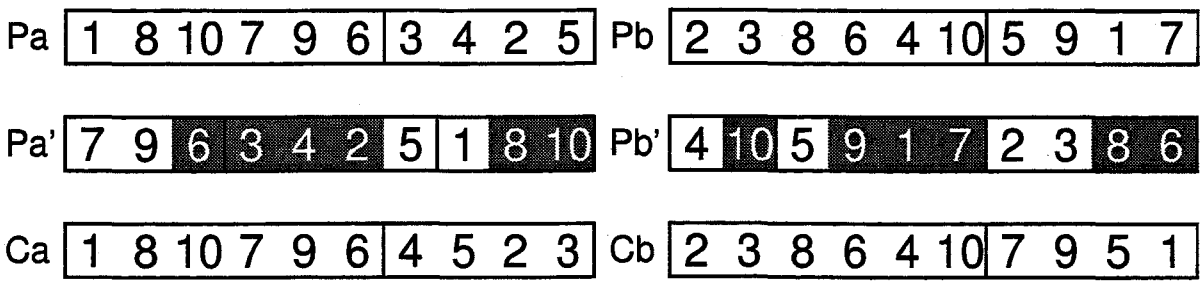


図 2.4: 手法1の交叉

Figure 2.4: The strategy of crossover for the method 1

算子では一方の親となる個体の染色体の交叉点直前の遺伝子座の遺伝子と同じ遺伝子を持つ遺伝子座をもう一方の親から探し，その次の遺伝子座を起点としていることである．図2.4でPaの交叉点直前の遺伝子座の遺伝子6をPbから探し，その直後の遺伝子座を先頭にして並べ変えたものがPb'である．同様にPa'は，Pbの交叉点直前の遺伝子座の遺伝子10をPaから探し，その直後の遺伝子座を先頭にして並べ変えたものである．CaはPaの前半部とPb'からPaの前半部にある遺伝子を取り除いたものから，CbはPbの前半部とPa'からPbの前半部にある遺伝子を取り除いたものから生成する．

突然変異

この手法での個体は同じ遺伝子を重複して持つことができないため，少なくとも二つの遺伝子座で同時に突然変異を起こす必要がある．そこで，この手法では図2.5に示す突然変異を用いた．まず，突然変異を起こす個体を集団からランダムに選択し，次に突然変異点をランダムに二箇所選択する．図2.5では5番目の遺伝子座の直前と8番目の遺伝子座の直後に突然変異点が設定されており，これらの二つの突然変異点に挟まれている遺伝子の並びを逆転させることで突然変異を起こした．

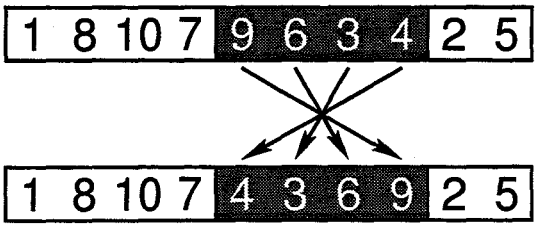


図2.5: 手法1の突然変異
Figure 2.5: The strategy of mutation for the method 1

2.2.2 コード化を順序表現とした手法(手法2)

コード化

この手法では致死遺伝子の生成を抑制するためにコード化を図2.6に示す順序表現(リストの何番目にある圃場を訪れるかを表す)とした．図2.6の染色体は遺伝子長が10，すなわち圃場数が10の時のもので，各遺伝子座は巡回順序を示し，各遺伝子は$1 \leq g(x) \leq L-(x-1)$

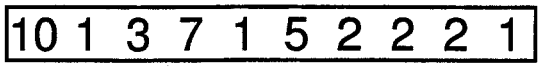


図 2.6: 手法 2 の染色体
Figure 2.6: A chromosome of the method 2

の値をとる．この染色体を持つ個体の遺伝子型と表現型は異なっているため，解の候補となる経路を得るためにはデコードする必要がある．デコードの方法は次のとおりである．

手順 1 $i = 1$，$l_1(x) = x$ に初期化する．

手順 2 $p(i) = l_i(g(i))$ とする．

手順 3 $i = L$ なら終了，そうでなければ手順 4 へ進む．

手順 4 $j = 1$ に初期化する．

手順 5 $j = g(i)$ なら手順 6 へ，そうでなければ $l_{i+1}(j) = l_i(j)$，$j = j + 1$ として 5 へ進む．

手順 6 $j = L - (i - 1)$ なら $i = i + 1$ として手順 2 へ，そうでなければ $l_{i+1}(j) = l_i(j + 1)$，$j = j + 1$ として手順 6 へ進む．

ここで

$g(x)$: x 番目の遺伝子座の遺伝子の値

L : 遺伝子長 (圃場数)

$p(x)$: 表現型で x 番目の値すなわち x 番目に訪れる圃場の番号

$l_i(x)$: i 番目の遺伝子座のデコードに用いるリストで x 番目に現れる圃場の番号

である．

図 2.7 は図 2.6 の個体をデコードする様子を示したものである．まず，遺伝子型での 1 番目の遺伝子が 10 であるため，圃場リストの 10 番目に現れる番号 10 を表現型の 1 番目とし，リストから 10 番目の番号を取り除いて更新する．この時，取り除いた番号以降に現れる番号を順次繰り上げる．ここでは取り除いた番号はリストの最後であるため，リストは 1, 2, 3, 4, 5, 6, 7, 8, 9 となる．次に，2 番目の遺伝子が 1 であるため，圃場リストの 1 番目に現れる番号 1 を表現型の 2 番目とし，同様にリストから 1 番目の番号を取り除いて更新し，2, 3, 4, 5, 6, 7, 8, 9 となる．この操作を繰り返し，この個体の表現型である巡回経路，10, 1, 4, 9, 2, 8, 5, 6, 7, 3 を得ることができる．

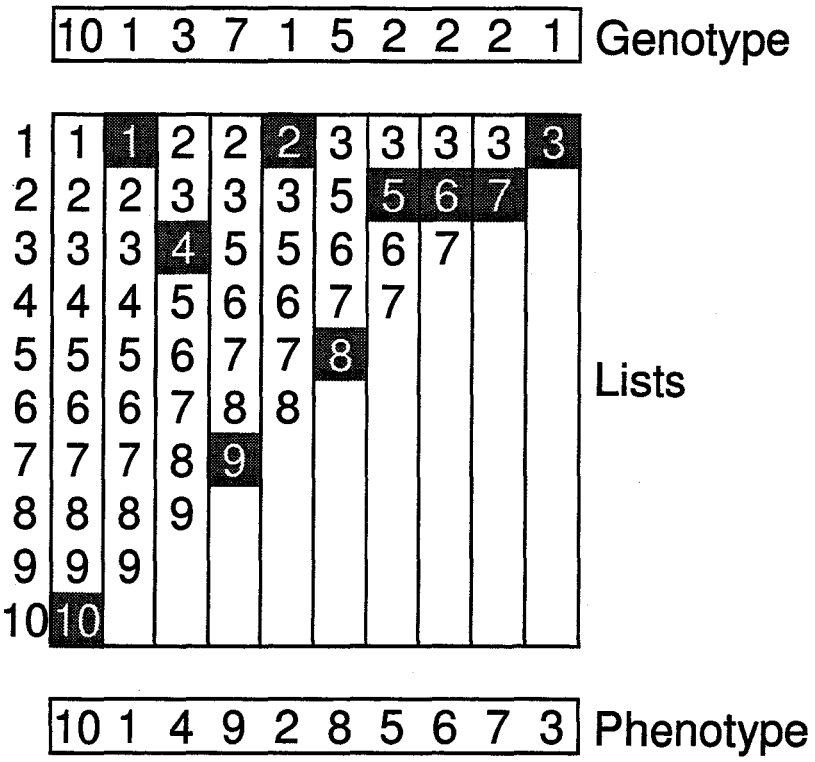


図 2.7: デコード
Figure 2.7: Decoding

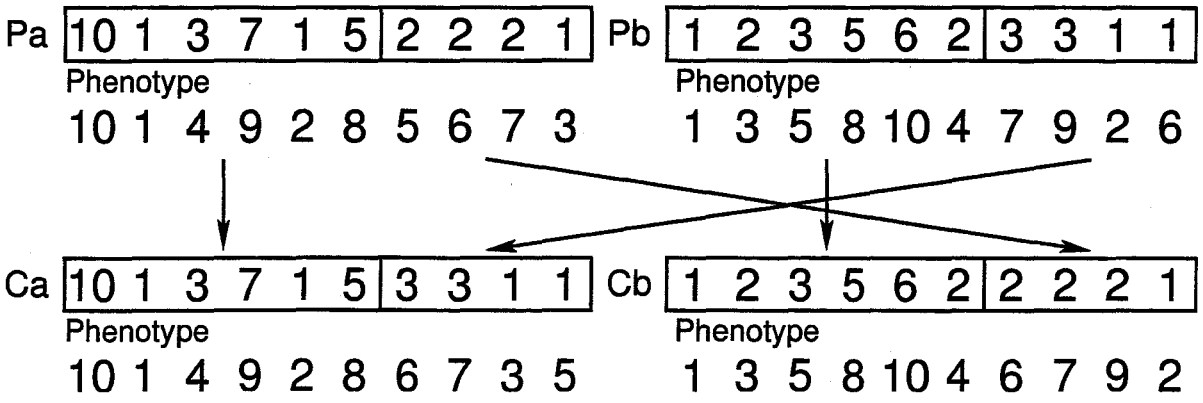


図 2.8: 手法 2 の交叉
Figure 2.8: The strategy of crossover for method 2

交叉

交叉は図2.8に示す一点交叉を用いた．この手法では各遺伝子座における遺伝子の値の範囲が遺伝子座の位置に依存するコード化を用いている．したがって，このコード化の方法による個体の染色体から一点交叉により生成したものも，各遺伝子座の遺伝子の値の範囲は遺伝子座の位置に依存したものとなる．そのため，図2.8のCa, Cbともに表現型にデコードするとTSPの条件を満たした経路となる．

突然変異

この手法での個体の遺伝子は，各遺伝子座の位置に依存した範囲を満たしていれば致死遺伝子とならないため，同じ遺伝子を重複して持つことができる．したがって手法1の様に少なくとも二つの遺伝子座で突然変異を起こさなければならない，といった制限はない．そこでこの手法では図2.9に示すように，ランダムに選択された個体の染色体から突然変異を起こす遺伝子を一箇所ランダムに選択し，その遺伝子を遺伝子座の位置に依存した範囲を満たす他の値に変更することで突然変異を起こした．

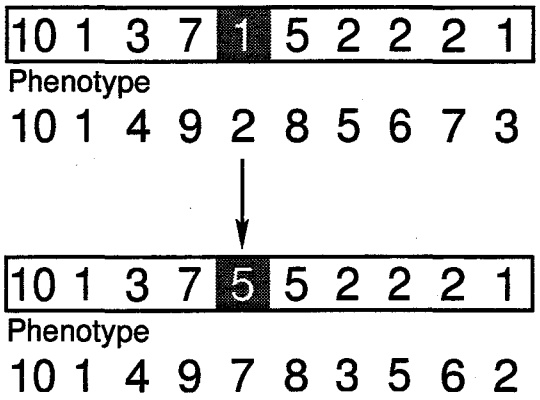


図 2.9: 手法2の突然変異
Figure 2.9: The strategy of mutation for method 2

2.2.3 アルゴリズム

手法1と手法2の相違点については前述した．ここでは両手法に共通する部分および計算の流れについて説明する．

まず，初期世代となる個体の染色体をランダムに一定数(個体数：N_p)生成し，各個体の適応度を計算して適応度の高いものから順位付けする．適応度の計算は以下のように行なう．

$$E = \sum_{i=1}^{N_f-1} d_{(i,i+1)} + d_{(1,N_f)} \tag{2.1}$$

$$F = \frac{1}{E} \tag{2.2}$$

ここで

E：個体の評価

N_f：圃場数

$d_{(i,j)}$：i番目に訪れる圃場とj番目に訪れる圃場の間の距離

F：個体の適応度

である．式(2.1)により得られるEは個体の表す巡回経路の総移動距離であり，式(2.2)によりEの逆数を求めて個体の適応度とした．

次に，これらの集団に遺伝的操作を行ない次世代へと進化させる．進化の手順は以下の通りである．

手順1 適応度の高いものから一定数(複製数：N_r)の個体を複製し，適応度の低い個体と入れ換える．

手順2 集団からランダムに一対の染色体を選び，各手法の交叉方法にしたがって交叉を行なう．この操作を一定数(交叉数：N_c)繰り返す．

手順3 これらの集団の各遺伝子に対して一定の確率(突然変異率：P_m)で突然変異を起こす．したがって，突然変異を起こす回数は$(P_m \cdot N_f \cdot N_p/100)$回となる．

手順4 各個体の適応度を計算して順位付けし，次世代の集団とする．

この進化の前後の集団の中でそれぞれ最も適応度の高い個体の適応度を比較し，進化後のものが低い場合には，進化後の最も適応度の低い個体を進化前の最も適応度の高い個体で

置き換えるエリート保存戦略を用いた．これにより，突然変異によって最優秀個体が破壊されることを防ぐことができる．

この進化を一定数 (世代数 : N_g) 繰り返して計算を終了し，最終世代の集団の中で適応度の最も高い個体を解とする．

2.3 手法1と手法2の性能比較

両手法の性能を比較するために，最適解が既知である問題をそれぞれの手法で解いた [7]．比較のために用いた問題は図2.10に示すように，同心二重円上に15度間隔で位置する48個の点の最短巡回経路を求めるものである．この問題では，半径比 $r/R \leq C$ の時は図2.11(a)，$r/R \geq C$ の時は図2.11(b) が最短巡回経路となり，$r/R = C$ の時は両巡回経路の距離は等しくなる．ここで，同心二重円上に存在する圃場数が $N_f (N_f = 4N, N$: 自然数) である時

$$C = \{1 - \sin(2\pi/N_f)\}/\{1 + \sin(2\pi/N_f)\} \tag{2.3}$$

である．

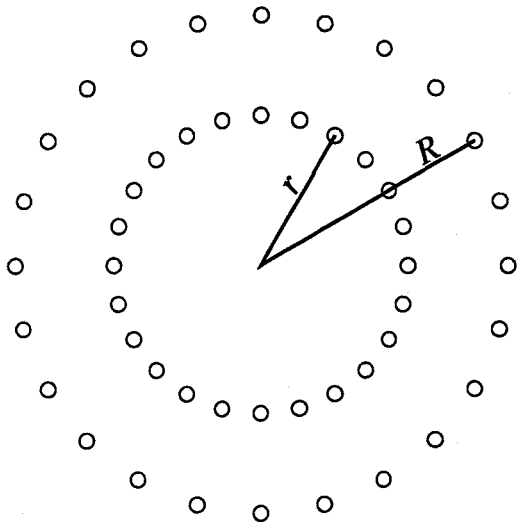
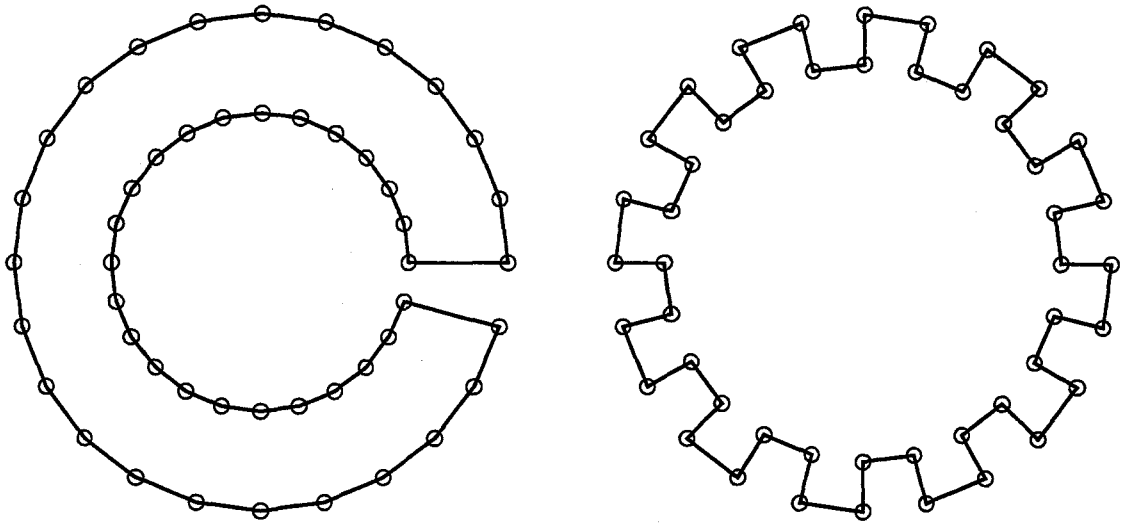


図 2.10: 圃場の位置
Figure 2.10: The field position



(a) $r/R \leq C$ (b) $r/R \geq C$

図 2.11: 最短経路
Figure 2.11: The shortest route

2.3.1 計算条件

計算に用いた圃場の位置はR = 50m, r = 30mで，最短巡回経路は図2.11(a)となり，この時の総移動距離は520.40mとなる．手法1，手法2ともに表2.1に示すとおり，N_g, N_p, N_r, N_cについては固定し，P_mを0.5%から3.0%まで0.5%刻みで変化させて計算を行なった．両手法とも各パラメータの組み合わせで50回ずつ計算し，得られた解を比較した．

表 2.1: パラメータ
Table 2.1: Parameters

最終世代数:N_g	1000
個　体　数:N_p	200
複　製　数:N_r	30
交　叉　数:N_c	95
突然変異率:P_m(%)	0.5, 1.0, 1.5, 2.0, 2.5, 3.0

2.3.2 計算結果

図2.12に手法1の，図2.13に手法2の突然変異率と解の平均値および95%信頼区間の関係を示す．図2.12，2.13に示すように，手法1では$P_m = 0.5\%$，手法2では$P_m = 2.0\%$で解の平均値が最小となっている．また，これらの図から各手法での最適な突然変異率の値を用いた時の結果を比較すると，手法1の方が優れていることが判明した．

表2.2に両手法の各計算条件での最良解 (Minimum)，平均値 (Average)，最悪解 (Maximum) を示す．手法1では$P_m = 0.5\%$で，手法2では$P_m = 3.0\%$で最良解が得られている．この時の巡回経路を図2.14に示す．図2.14(a) は$r/R < 0.77$の時の最短巡回経路であり，手法1では最適解が得られたのに対し，手法2での最良解の経路は図2.14(b) に示すように，経路の交差が数多く見られ，最短経路とはかけ離れた状態である．これらの経路が得られた時の世代数と各世代の最優秀個体の移動距離の関係を図2.15，図2.16に示す．図2.15，図2.16が示すように，手法1，手法2ともにランダムに初期世代を生成するため，初期世代の最優秀個体の距離はほぼ同じくらいである．手法1では極端に局所解に陥ることなく世代が進行しており，710世代で最適解に到達している．これに対して手法2では比較的初期の世代では大きく改善されているが，100世代を超えた辺りからは徐々にグラフの傾きが緩やかになっている．これは手法2の交叉の場合，遺伝子型では交叉点以降の部分も親の遺伝子を継承しているが，これを表現型に変換すると交叉点以降では親の形質を継承しなくなるためであると考えられる．そのため，図2.17のように交叉点が前の方にあると親の形質をほとんど受け継がず，ランダムに個体を生成したのと同様の結果となる．したがって，集団に適応度の高い個体が現れていない比較的初期の世代では，急速に適応度の高い個体が出現するが，ある程度集団の適応度が高くなるとさらに適応度の高い個体はほとんど現れなくなり，局所解に陥りやすくなる．

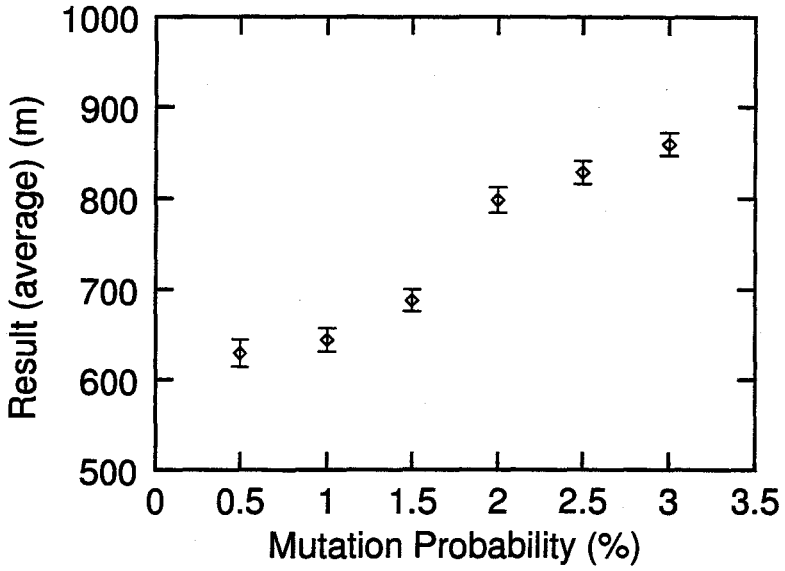


図 2.12: 突然変異率と解の関係 (手法 1)

Figure 2.12: Relation between mutation probability and result (Method 1)

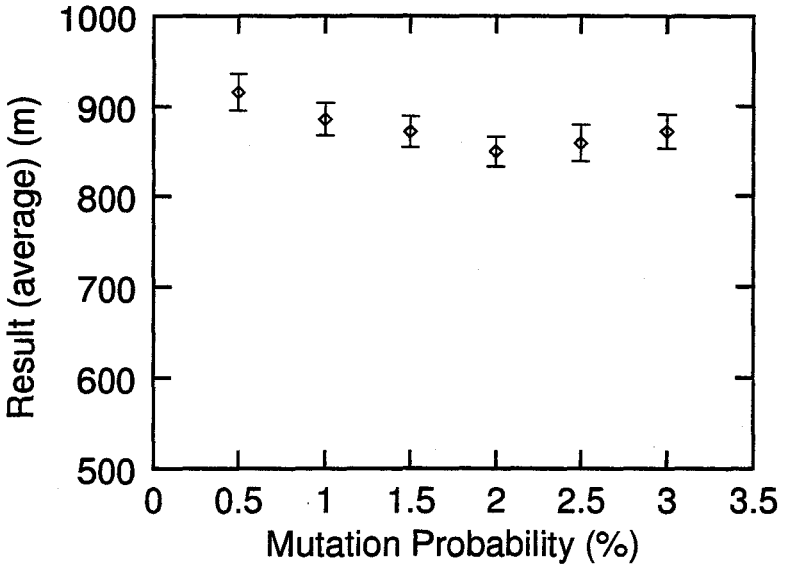


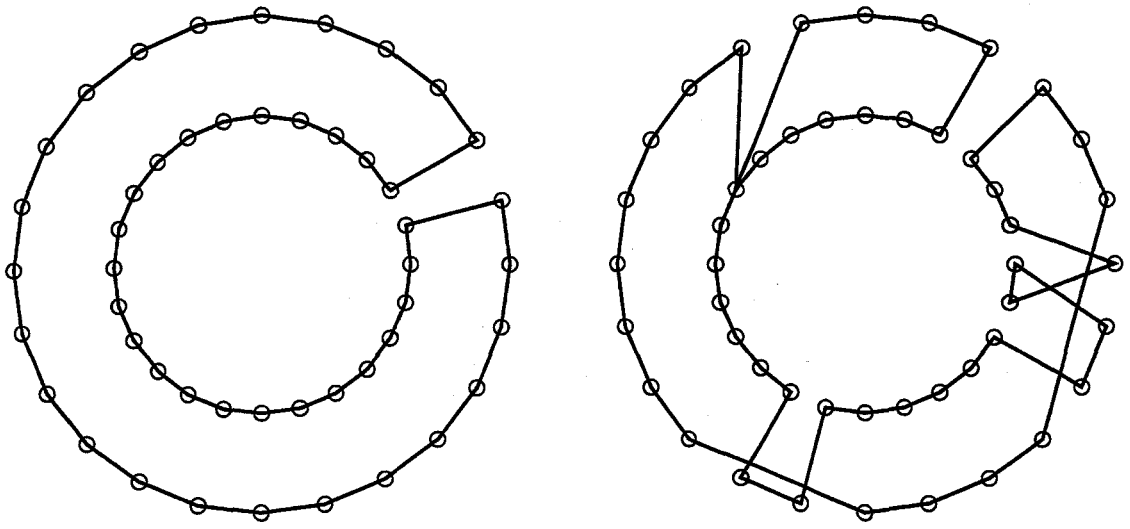
図 2.13: 突然変異率と解の関係 (手法 2)

Figure 2.13: Relation between mutation probability and result (Method 2)

表 2.2: 計算結果
Table 2.2: Results of calculation

単位：m

$P_m(\%)$		0.5	1.0	1.5	2.0	2.5	3.0
Method1	Minimum	520.40	525.15	551.02	726.85	754.39	762.54
	Average	628.95	644.23	687.81	798.38	828.59	859.48
	Maximum	716.64	707.86	767.67	933.76	953.40	988.31
Method2	Minimum	802.56	766.21	719.04	725.77	728.32	698.30
	Average	916.01	885.72	871.52	849.44	858.72	871.95
	Maximum	1108.84	1023.83	1011.17	978.67	1117.48	999.79



(a) Method1 (b) Method2

図 2.14: 各手法で得られた最良解
Figure 2.14: The best result of each method

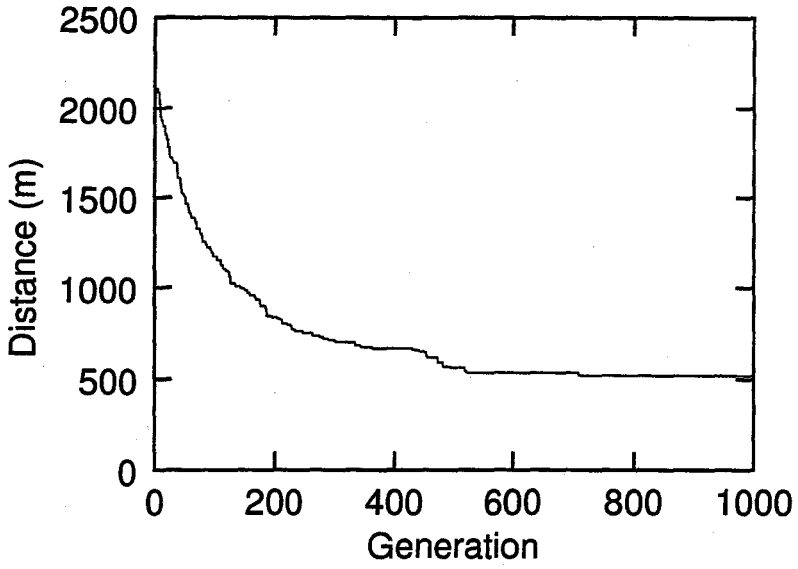


図 2.15: 手法 1 の最優秀個体の移動距離の推移
Figure 2.15: Change of distance of the best individual in method 1

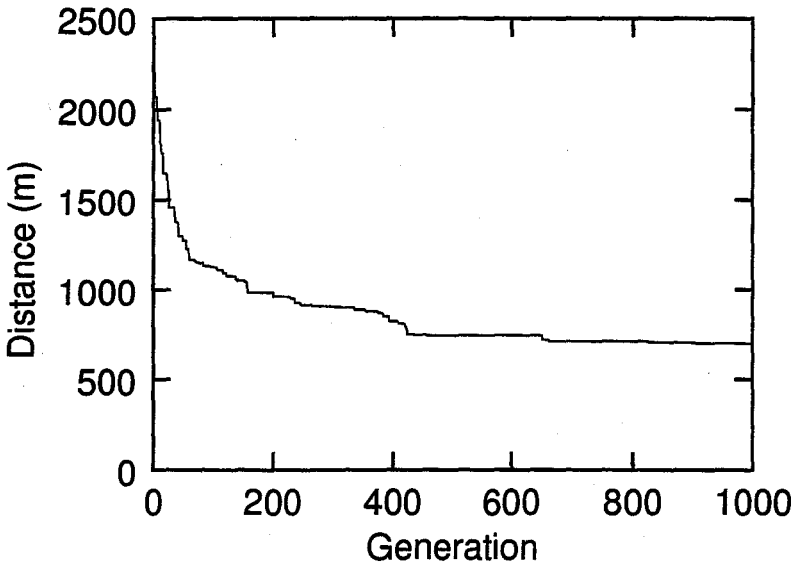


図 2.16: 手法 2 の最優秀個体の移動距離の推移
Figure 2.16: Change of distance of the best individual in method 2

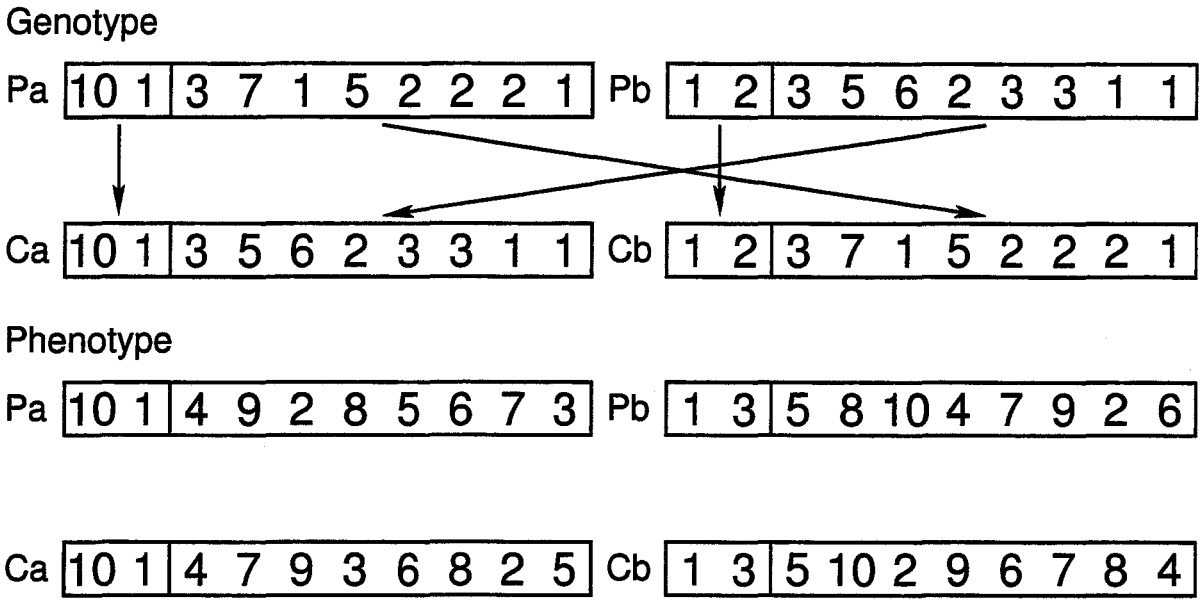


図 2.17: 手法2で子が親の形質を継承しない例
Figure 2.17: An example that children inherit little from parents in method 2

2.4 適応度に応じた選択を行なう手法

前節の結果より，手法1の方が手法2より優れていることが明らかになったが，表2.2に示すように，最も解の平均値が優れていた$P_m = 0.5\%$の時でも最悪解が716.64mと最適解からかけ離れた結果を得ることがあった．そこで，手法1をもとに次節に説明する変更を加えた手法で計算を行なった．

2.4.1 変更点

手法1では，複製する個体の選択は適応度の高い一定数(N_r)の個体に固定されており，集団を進化させる時にこれらの個体を適応度の低い個体と入れ換えた．この操作を行なうことにより，交叉を行なう個体が選択される確率は適応度により変化することとなる．すなわち，複製後の集団では複製前の集団での適応度が上位のN_r個の個体はそれぞれ二つずつ存在し，下位のN_r個の個体は存在しないため，複製後の集団からランダムに交叉を行なう個体を選択することは複製前の集団から適応度が上位であるN_r個の個体，中位の$N_p - 2N_r$個の個体，下位のN_r個の個体を2：1：0の確率で選択することと等価となる．しかし，この方法では下位のN_r個の個体は全て淘汰され，また交叉では，上位のN_r

個の個体，中位の $N_p - 2N_r$ 個の個体がそれぞれ適応度に関わらず同じ確率で選択されることになる．そこで，適応度が下位の個体でも適応度に応じて次世代の個体として選択され，また交叉を行なう個体も適応度に応じて選択されるようにルーレット選択 (roulette selection) およびランク選択 (rank selection) を採用した [8].

ルーレット選択では各個体は適応度に比例した確率で選択され，ランク選択では全個体をランク付けしランクに応じてある一定の確率で選択される．ルーレット選択，ランク選択で各個体が選択される確率を式 (2.4)，(2.5) に示す．

$$P_{s_{ro}}(x) = \frac{F_x}{F_{sum}} \tag{2.4}$$

$$P_{s_{ra}}(x) = \frac{N_p - (x-1)}{N_{sum}} \tag{2.5}$$

ここで

$P_{s_{ro}}(x)$: x 番目に適応度の高い個体がルーレット選択により選択される確率

$P_{s_{ra}}(x)$: x 番目に適応度の高い個体がランク選択により選択される確率

F_x : x 番目に適応度の高い個体の適応度

であり，F_{sum}, N_{sum} は

$$F_{sum} = \sum_{i=1}^{N_p} F_i \tag{2.6}$$

$$N_{sum} = \sum_{i=1}^{N_p} i \tag{2.7}$$

に従う．ルーレット選択である個体が選択される確率は，集団の適応度の和と個体の適応度に依存するのに対して，ランク選択ではある個体の集団でのランクにのみ依存する．したがってランク選択で，あるランクの個体が選択される確率はどの世代でも同じになる．

また，エリート保存戦略により，集団の多様性が失われるのを防ぐために，同一世代での同一個体の数を 20 に制限し，制限を超えた場合は一個体だけを残し，その他は初期世代と同様の方法でランダムに生成したものと入れ換えた．

2.4.2 アルゴリズム

前述の変更に伴い，進化の手順も次のように変更した．

手順 1 ルーレット選択，あるいはランク選択により適応度に応じて次世代に生き残る個体を N_r 個選択する．

手順 2 集団からルーレット選択，あるいはランク選択により一対の染色体を選び，交叉を行なう．この操作を一定数 (交叉数 : N_c) 繰り返す．

手順 3 これらの集団の各個体に対して一定の確率 (突然変異率 : P_m) で突然変異を起こす．したがって，突然変異を起こす回数は $(P_m \cdot N_p/100)$ 回となる．

手順 4 各個体の適応度を計算して次世代の集団とする．この時，エリートは保存する．また，適応度が等しい個体を同一個体とみなし，同一個体が 20 を超えた時は一個体を残し，その他はランダムに生成する．

2.4.3 計算条件および計算結果

ルーレット選択とランク選択による計算結果を比較するために，前節と同様に半径がそれぞれ 30m，50m の同心円上に 15 度間隔で位置する 48 個の点の最短巡回経路を求めた．ルーレット選択，ランク選択ともに表 2.3 に示すように，N_g, N_p, N_r, N_c については固定し，P_m を 0.5%から 3.0%まで 0.5%刻みで変化させて計算を行なった．なお，複製数が変化しているのは，進化の手順を変更したため個体数，複製数，交叉数の関係が $N_p = N_r + 2N_c$ となっているからである．また，突然変異率の扱いは，前節では遺伝子座の数に対する値であったが，ここでは個体に対する値に変更した．前節の手法 1 では二つの遺伝子座の間の遺伝子の並びを逆転させることで，また手法 2 ではある遺伝子座の遺伝子の値を変更することで突然変異としているため，突然変異率は手法 1 ではある個体で突然変異が起こる確率，手法 2 ではある遺伝子座で突然変異が起こる確率として扱うべきであるが，前節ではこれらの手法を比較するために突然変異率の扱いを手法 2 のものに統一した．本節の手法では前節の手法 1 の突然変異の方法を用いているため，突然変異率はある個体で突然変異が起こる確率として扱った．表 2.3 の各パラメータの組み合わせで 50 回ずつ計算し，得られた解を比較した．

表 2.3: パラメータ
Table 2.3: Parameters

最終世代数:N_g	1000
個　体　数:N_p	200
複　製　数:N_r	$N_p - 2N_c$
交　叉　数:N_c	80
突然変異率:P_m(%)	0.5, 1.0, 1.5, 2.0, 2.5, 3.0

図 2.18 にルーレット選択，図 2.19 にランク選択を用いた時の突然変異率と解の平均値および 95%信頼区間の関係を示す．図 2.18，2.19 に示すように，ルーレット選択では $P_m = 0.5\%$，ランク選択では $P_m = 2.5\%$で解の平均値が最小となっている．また，これらの図から各手法での最適な突然変異率の値を用いた時の結果を比較すると，ランク選択の方が優れていることが判明した．

表 2.4 にルーレット選択とランク選択を用いた時の各計算条件での最良解，平均値，最悪解を示す．ルーレット選択では $P_m = 0.5\%$で最良解が，ランク選択では全ての P_m で最適解が得られている．ランク選択を用いたものでは，ランダムに選択していたものに比べて，最良解，平均値，最悪解がともに改善されているが，ルーレット選択を用いたものでは逆に悪化している．これは，適応度にばらつきが少ない時ルーレット選択ではランダム選択と同様になり，優秀な個体が選択されやすくならないのに対して，ランク選択では優秀な個体が確実に選択されやすくなるためであると考えられる．

表 2.4: 計算結果
Table 2.4: Results of calculation

単位 : m

P_m(%)		0.5	1.0	1.5	2.0	2.5	3.0
Method1	Minimum	681.61	768.06	828.95	833.51	767.38	755.65
Roulette	Average	797.75	937.35	951.91	973.10	987.56	978.04
Selection	Maximum	981.38	1092.36	1101.15	1163.41	1288.44	1214.50
Method1	Minimum	520.40	520.40	520.40	520.40	520.40	520.40
Rank	Average	614.48	605.99	595.60	594.54	589.95	599.96
Selection	Maximum	713.20	671.59	667.19	674.11	668.38	666.98

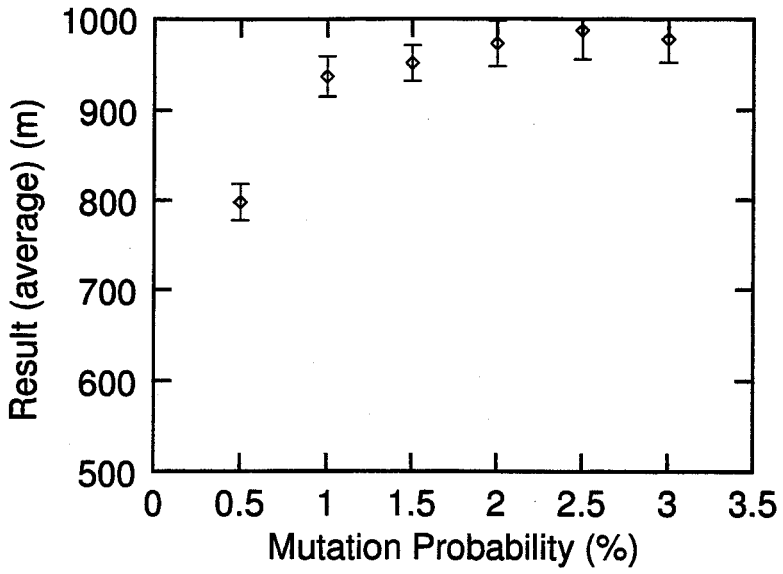


図 2.18: 突然変異率と解の関係 (ルーレット選択)
Figure 2.18: Relation between mutation probability and result (Roulette Selection)

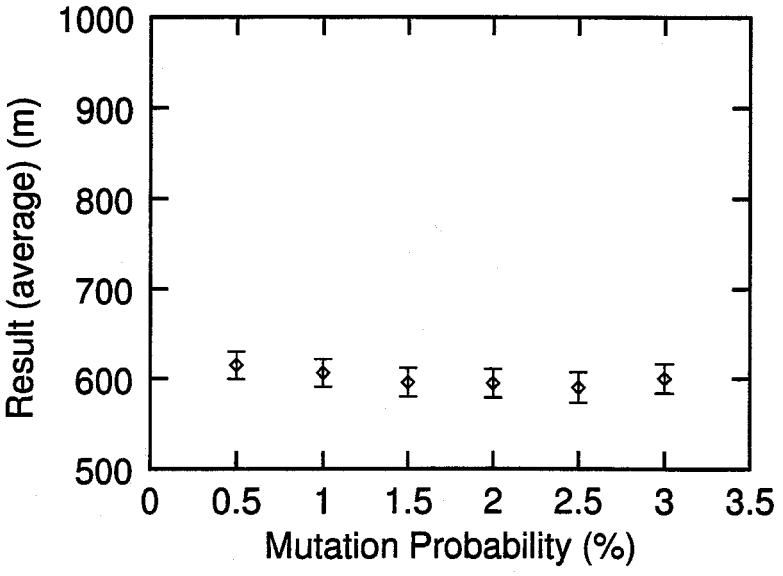


図 2.19: 突然変異率と解の関係 (ランク選択)
Figure 2.19: Relation between mutation probability and result (Rank Selection)

以上の結果より手法1にランク選択を組み合わせた手法が有効であることが明らかになった．表2.4よりランク選択を用いた時，突然変異率P_mを2.0%および2.5%に設定することで解の平均値が小さくなることが示されている．そこで突然変異率をこれらの値に設定し，交叉数を変化させることで交叉数と解の関係を調べた．各パラメータを表2.5に示す．各計算条件で50回ずつ計算し，最良解，解の平均値，最悪解を比較した．表2.6に各計算条件での計算結果を示す．各計算条件で最短経路が得られており，また，各計算条件の間で大きな差がないことが判明した．

表2.5: パラメータ
Table 2.5: Parameters

最終世代数:N_g	1000
個　体　数:N_p	200
複　製　数:N_r	$N_p - 2N_c$
交　叉　数:N_c	70, 80, 90
突然変異率:P_m(%)	2.0, 2.5

表2.6: 計算結果
Table 2.6: Results of calculation

単位：m

P_m(%)	2.0			2.5		
N_c	70	80	90	70	80	90
Minimum	520.40	520.40	520.40	520.40	520.40	520.40
Average	581.66	594.54	579.68	572.02	589.95	574.18
Maximum	664.66	674.11	658.82	665.27	668.38	668.25

2.5 計算結果および考察

2.5.1 保管庫と9箇所の圃場の巡回経路の最適化

各手法，および選択の方法を検討した結果，手法1にランク選択を組み合わせたものが最も有効であることが明らかになった．そこでこれを実際の圃場データを用いた最適化に適用した．計算に用いた圃場データは京都府福知山市西中筋東部地区で，京都府が行なう水田農業大区画圃場整備事業の対象となっている．図2.20に大区画化された圃場の写真を示す．図2.21はこの地区の地図で1が保管庫，2から10が圃場である．これらの10地点の相互間の距離をデータとして与えた．表2.7に示したこれらのデータのうちMで示したものについては，それらの圃場間を直接移動することはないものとして大きな数字を与えた．これは例えば，保管庫1から圃場3に最短距離で移動する時には必ず圃場2を通るからである．これは樋口がニューラルネットワークによりこれらの10地点の最短巡回経路を求めた時に用いた方法[9]で，これにより計算回数を減らすことができる．このニューラルネットワークによる手法と比較するために，本研究でも距離データを同様に扱った．計算に用いたパラメータは，前節の計算結果で最も解の平均値が小さかった組み合わせ，最も大きかった組み合わせである表2.8に示すものとし，それぞれの組み合わせで50回ずつ計算を行なった．

表2.7: 圃場間の距離
Table 2.7: The distance matrix between fields

単位：m

Field No.	1	2	3	4	5	6	7	8	9	10
1	-	250	M	M	M	625	M	M	125	300
2	250	-	150	M	M	M	M	M	275	550
3	M	150	-	325	400	425	M	M	M	M
4	M	M	325	-	330	125	650	M	M	M
5	M	M	400	330	-	450	575	M	M	M
6	625	M	425	125	450	-	550	450	550	M
7	M	M	M	650	575	550	-	675	M	M
8	M	M	M	M	M	450	675	-	M	M
9	125	275	M	M	M	550	M	M	-	425
10	300	550	M	M	M	M	M	M	425	-

図 2.20: 対象圃場 (福知山市西中筋東部地区)
Figure 2.20: The fields in Nishinakasuji-Tobu of Fukuchiyama city

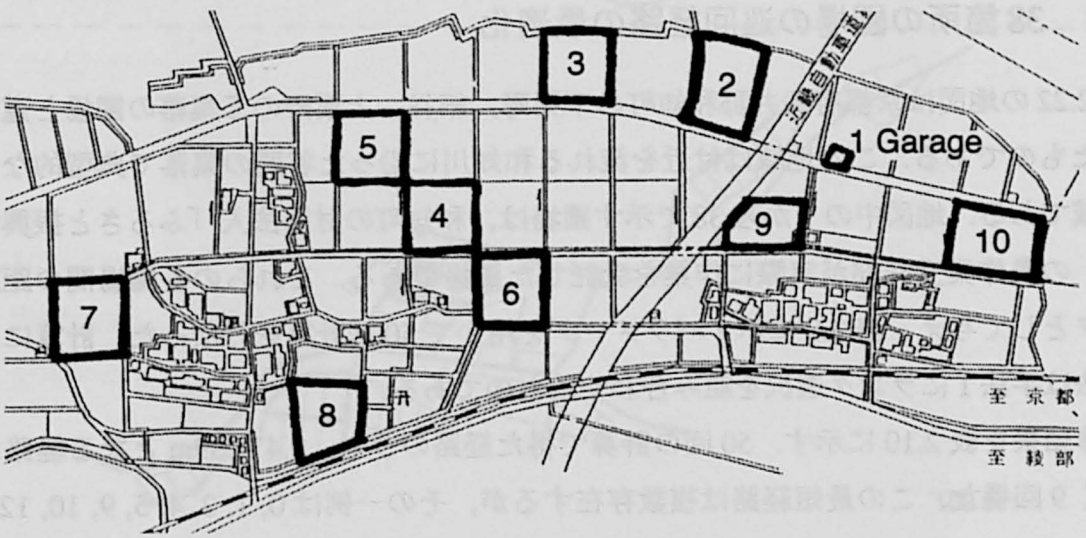


図 2.21: 圃場の位置 (福知山市西中筋東部地区)
Figure 2.21: The location of fields in Nishinakasuji-Tobu of Fukuchiyama city

表 2.8: パラメータ
Table 2.8: Parameters

最終世代数:N_g	200	200
個 体 数:N_p	100	100
複 製 数:N_r	$N_p - 2N_c$	$N_p - 2N_c$
交 叉 数:N_c	35	40
突然変異率:P_m(%)	2.5	0.5

それぞれのパラメータの組み合わせで計算した結果，解は全て4,030mとなった．総移動距離が4,030mとなる経路は複数あるが，これは機械が通ることのできる道が限られているためで．得られた経路のうちの一つは1, 10, 2, 3, 4, 5, 7, 8, 6, 9, 1を順に訪れるものであったが，1, 2, 3, 4, 5, 7, 8, 6, 9, 10, 1であっても機械が通る道路は全く同じになるためである．ニューラルネットワークにより得られた解も4,030mであったが，ニューラルネットワークではパラメータの決定が難しく，適当でないパラメータを与えると収束しないという問題があった．これに対して本研究で用いたGAでは，前節の計算結果が最も悪かったパラメータの組み合わせでも全て4,030mという結果を得ることができた．

2.5.2 38箇所の圃場の巡回経路の最適化

図2.22の地図は京都府船井郡和知町の下粟野，細谷，上粟野の三集落の圃場と道路を示したものである．この地域は付近を流れる和知川に沿った谷間の集落で典型的な中山間地域である．地図中の1から38で示す圃場は，和知町の財団法人「ふるさと振興センター」の農作業受託部が実際に作業を受託した圃場である．これらの各圃場間の距離をデータとして与え，表2.9に示すパラメータを用いて50回計算を行なった．計算に用いた手法は手法1にランク選択を組み合わせたものである．

計算結果を表2.10に示す．50回の計算で得た経路のうち，9,472.39mとなる経路が最も短く9回得た．この最短経路は複数存在するが，その一例は0, 1, 2, 4, 5, 9, 10, 12, 16, 18, 26, 27, 37, 35, 36, 33, 34, 31, 29, 30, 28, 32, 23, 25, 24, 22, 21, 17, 19, 20, 14, 15, 11, 13, 8, 6, 7, 3というものであった．また，最悪解は9,541.36mであり，最良解との誤差は1%以下であった．この最悪解は，50回の計算の中で1回しか出現しておらず，この解を除いた49回の計算結果は9,489.64m以下となり，最良解との誤差は0.2%以下であった．

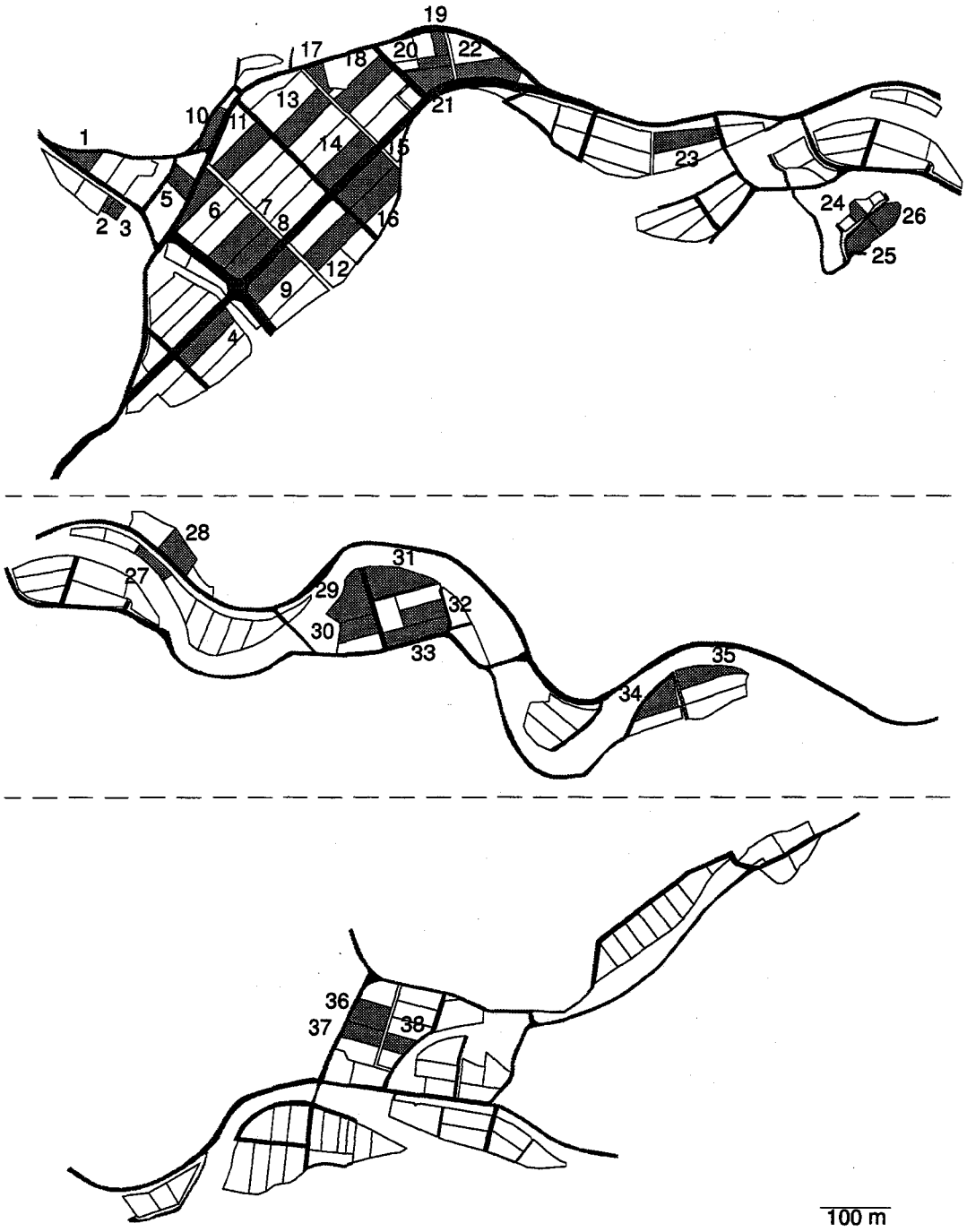


図 2.22: 圃場の位置 (和知町)
Figure 2.22: The location of fields in Wachi

表 2.9: パラメータ
Table 2.9: Parameters

最終世代数:N_g	1000
個 体 数:N_p	200
複 製 数:N_r	$N_p - 2N_c$
交 叉 数:N_c	70
突然変異率:$P_m(\%)$	2.5

表 2.10: 計算結果
Table 2.10: Results of calculation

Minimum (m)	Average (m)	Maximum (m)
9,472.39	9,479.43	9,541.36

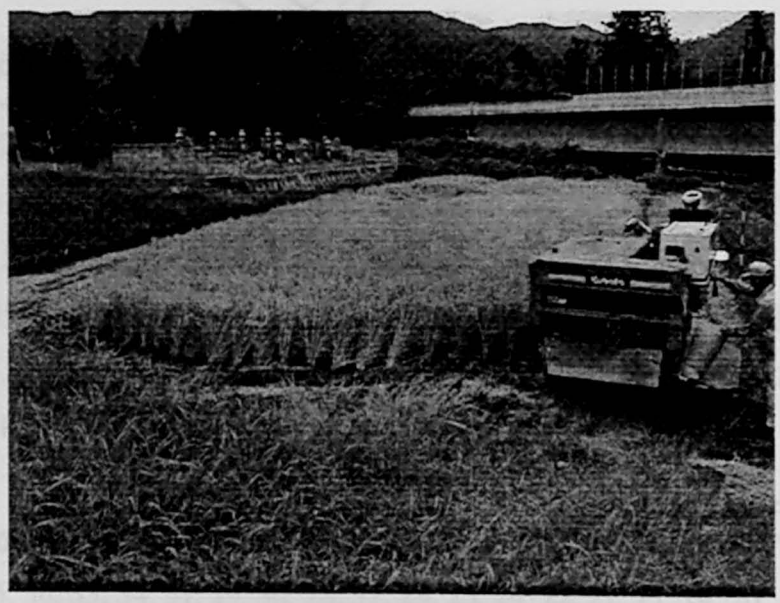


図 2.23: 対象圃場 (和知町)
Figure 2.23: The fields in Wachi

この計算に用いた圃場での作業を受託している農作業受託部では，和知町全域で作業を受託しており，その圃場の数は500筆強にも及ぶ．また，それらの圃場は図2.23に示すように狭いものが多く，川などを挟んで点在しているため最適な移動経路を得るのは困難である．そこで，本研究で用いた手法により作業経路を決定するのは作業効率を改善するために有効である．

また，ニューラルネットワークを用いた手法では，圃場数が変化すると，同じパラメータでは収束しなかったが，本研究の手法では同じパラメータで対応できた．この点でも本研究の手法が有効であることが確認できる．

2.6 まとめ

一台の農業機械が，割り当てられた圃場を順に訪れ作業する時の最短経路を，GAを用いてTSPとして解く手法について，コード化の方法，選択の方法，各パラメータの違いにより解を比較し，その結果，以下のことが明らかになった．

1. コード化が経路表現，順序表現となる二つの手法を比較すると，前者の方が優れていた．

2. 解の精度を高めるために適応度に応じた選択の方法として，ルーレット選択およびランク選択を導入し，両者の計算結果を比較したところ，後者が有効であることを確認した．

3. さまざまなパラメータで計算した結果，ニューラルネットワークを用いた手法ではパラメータの決定が難しく，適当でないパラメータを与えると収束しない，また圃場数が変化すると同じパラメータでは対応できないという問題点があったが，GAを用いた手法ではこれらの問題点を解消できることを確認した．

参考文献

[1] 山崎稔, 笈田昭, 樋口昌吾：ニューラルネットワーク手法によるトラクタの圃場巡回問題へのアプローチ, 農業機械学会関西支部報, 74, pp.99-100, (1993)

[2] D. Goldberg and R. Lingle, Jr. : Alleles, Loci, and the traveling salesman problem, Proc. of 1st Int. Conf. on Genetic Algorithms and Their Applications, pp.154-159, (1985)

[3] L. Davis : Applyling adaptive algorithms to epistatic domains, Proc. 9th Int. Joint Conf. on Artificial Intelligence, pp.162-164, (1985)

[4] I. M. Oliver, D. J. Smith and J. R. C. Holland : A study of permutation crossover operators on the Traveling Salesman Problem, Proc. of 2nd Int. Conf. on Genetic Algorithms, pp.224-230, (1987)

[5] J. Grefenstette, R. Gopal, B. Rosmaita and D. Van Gucht : Genetic algorithms for the traveling salesman problem, Proc. of 1st Int. Conf. on Genetic Algorithms and Their Applications, pp.160-168, (1985)

[6] K. Ohdoi, A. Oida and M. Yamazaki : Application of Genetic Algorithms to solve the scheduling problem in farm fields, Proc. of Int. Symposium on Automation and Robotics in Bioproduction and Processing, Vol. I, pp.235–242, (1995)

[7] K. Ohdoi, A. Oida and M. Yamazaki : Application of Genetic Algorithms to solve the working around tractor problem, Proc. of the Int. Agricultural Engineering Conference, Vol. II, pp.740–746, (1994)

[8] 大土井克明, 笈田昭, 山崎稔, 山下道弘：農作業計画の最適化に関する研究 (第 1 報) —作業経路の最適化—, 農業機械学会誌, 61(1), pp.91–97, (1999)

[9] 樋口昌吾：ニューラルネットワーク手法によるトラクタの圃場巡回問題へのアプローチ, 京都大学農学部卒業論文, (1993)

第3章　圃場割り当ての最適化

3.1　はじめに

第2章では作業をする圃場を割り当てられた機械の最適な作業圃場順序について，機械の圃場間の移動距離を最小にすることを目的としてGAにより解を求める方法を検討した．この方法では一台の機械が作業して回る圃場の順序について考えており，複数の機械に対応することはできない．そこで本章では複数の機械で作業する時の各機械への最適な圃場割り当てを行なう手法について考える．圃場割り当てを最適化する場合，作業適期，天候，水管理などの拘束条件を考慮する必要があるが，ここでは一日の実作業時間と割り当てられた圃場での作業にかかる時間との誤差を小さくすること，圃場間の移動距離を短くすることの二つの条件に焦点を当て最適化を行なった．これは少なくともこれらの二つを考慮しないと，実行可能な解が得られないためである．すなわち，これらの二つを評価の対象とせずに例えば作業適期や天候などの条件だけを考えた場合，一日の実作業時間の数倍の時間のかかる作業を行なわなければならない圃場割り当てや，圃場間の移動距離の長い圃場割り当てが最適解となってしまうからである．したがって，圃場割り当ての最適化を考える場合，これらの二つを考慮した上で他の拘束条件を評価の対象とする必要がある．

3.2　一台ずつ圃場を割り当てる手法(手法1)

第2章では，一台の機械が点在する圃場を順番に作業する時の最適な移動経路をTSPに単純化し，GAを用いて解を求めた．本章でも最適化の手法として組合せ最適化問題に適したGAを用いるが，複数の機械への圃場割り当てについて考えるため，コード化の方法や評価の方法を新たに考えた[1].

3.2.1 コード化

図3.1にこの手法で用いたコード化の方法で生成した個体の染色体を示す．この個体の遺伝子長は圃場数と一致し，各遺伝子座に圃場が対応している．図3.1(a)は機械番号1の機械が作業すべき圃場を決定する時の個体の染色体で，各遺伝子は0または1の値をとり，0がその遺伝子座に対応する圃場を1番の機械で作業しない，1は作業するを表す．したがって(a)は圃場数が10の時の個体の染色体で，1番の機械が2, 3, 5, 9の各圃場で作業することを表している．図3.1(b)は機械番号$n_m(n_m > 1)$の機械が作業すべき圃場を決定する時の個体の染色体である，各遺伝子は0，1，9の値をとる．これらの値のうち0，1が表すのは(a)と同様で，作業しない，するを表し，9はその遺伝子座に対応する圃場は$n_m - 1$番までの機械で作業をすることが既に決まっていることを表す．したがって(b)は圃場数が10の時の個体の染色体で，2, 3, 5, 9, の各圃場で作業する機械が$n_m - 1$番までの機械で作業することが決まっている状況で，n_m番の機械が1, 7, 10の各圃場で作業をすることを表している．

(a) | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

(b) | 1 | 9 | 9 | 0 | 9 | 0 | 1 | 0 | 9 | 1 |

図3.1: 手法1の染色体
Figure 3.1: A chromosome of the method 1

3.2.2 アルゴリズム

この手法では所有する機械に番号をつけ，番号の小さい機械から順に作業すべき圃場を決定し，全ての圃場がいずれかの機械に割り当てられるまで繰り返す．まず，一日目の1番の機械について，図3.1(a)にしたがう初期世代の個体を一定数(個体数：N_p)だけランダムに生成し，各個体の適応度を計算する．この集団に対して複製，交叉，突然変異などの遺伝的操作を行ない，集団を次世代へと進化させる．進化の手順は以下の通りである．

手順1 次世代の候補となる$N_p + 2N_c$個(N_c：交叉数)の個体を保存する配列を用意する．

手順2 集団の全ての個体を次世代の候補として複製する．

手順 3 集団から適応度に応じて一対の染色体を選び，交叉を行なう．この操作を一定数 (交叉数 : N_c) 繰り返し，次世代の候補となる個体を生成する．

手順 4 複製，交叉により生成した次世代の候補となる集団のうち，複製による個体に対して一定の確率 (突然変異率 : P_m) で突然変異を起こす．したがって，突然変異を起こす回数は $(P_m \cdot N_p/100)$ 回となる．

手順 5 次世代の候補となる各個体の適応度を計算して上位のものから一定数 (個体数 : N_p) を残し，次世代の集団とする．

これらの操作を一定回数 (最終世代数 : N_g) 繰り返し，1 番の機械が一日目に作業をする圃場を決定する．この時，作業する機械が決まっていない圃場，すなわち 1 番の機械に割り当てられていない圃場について，2 番の機械で一日の実作業時間内に作業を終了できれば，それらの圃場を 2 番の機械で作業するものとして計算を終了する．実作業時間内に作業を終了できない時は図 3.1(b) の個体を用いて同様の計算を行ない，2 番の機械で作業する圃場を決定する．これらの計算を終了条件，すなわち $n_m - 1$ 番の機械までに割り当てられていない圃場を n_m 番の機械で一日の実作業時間内に完了できる，を満たすまで繰り返して行い，全ての圃場について作業を行なう機械を決定する．ここで，所有機械の台数が 3 台であれば，実際には 1〜3 番の機械しか存在しないが計算上 4 番の機械は 1 番の機械の二日目の作業を，8 番の機械は 2 番の機械の三日目の作業を表すこととする．

3.2.3 遺伝的操作

遺伝的操作のうち，交叉は図 3.2 に示す一点交叉を用いた．本章の手法では進化の手順が第 2 章の手法と異なっているため，まず適応度に比例して個体を選択するルーレット選択により交叉を行なう個体を決定した．第 2 章で良好な結果を得たランク選択を用いた手法については後述する．選択された一対の個体に対してランダムに交叉点を決定し，交叉点以降の遺伝子を入れ換えて新たな個体を生成する．図 3.2 に示した個体は n_m 番 $(n_m \geq 2)$ の機械への圃場割り当てを行なう時のものであるが，初期世代生成の時に全ての個体で，$n_m - 1$ 番までの機械に割り当てられた圃場に対応する遺伝子座の遺伝子を 9 にしているため，これらの遺伝子座の遺伝子は交叉後も 9 となる．

突然変異は図 3.3 に示すように，ランダムに選ばれた個体からランダムに遺伝子座を選

び，その遺伝子座を反転，すなわち0であれば1，1であれば0に変化させ，9であればやり直しとした．

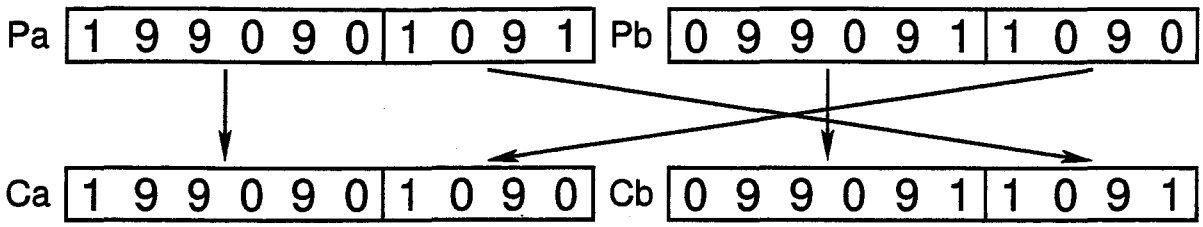


図 3.2: 手法1の交叉
Figure 3.2: The strategy of crossover for method 1

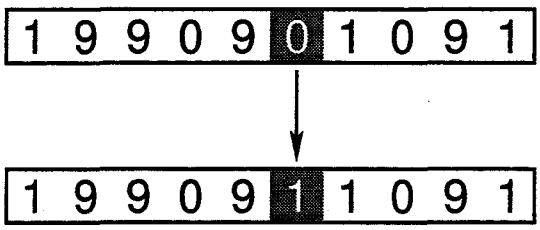


図 3.3: 手法1の突然変異
Figure 3.3: The strategy of mutation for method 1

3.2.4 適応度

各個体の適応度を決定する評価関数はGAの性能に大きく影響を与えるため，最適化の目的に応じて決定する必要がある．本手法での最適化の目的は，一日の実作業時間と割り当てられた圃場での所要作業時間の誤差を少なくし，かつ，各機械の圃場間の移動距離を少なくするというものであるため，評価関数には時間に関する評価と距離に関する評価の二つを導入した．

時間に関する評価は，各機械に割り当てられた圃場での所要作業時間と一日の実作業時間から，

$$E_t = |t - T|/T \tag{3.1}$$

により求める．ここで

E_t : 時間に関する評価

t : 割り当てられた圃場での所要作業時間

T : 一日の実作業時間

である．なお，$t > 2T$ の時，すなわち割り当てられた圃場での所要作業時間が一日の実作業時間の 2 倍を超えた時は，その個体を環境に適応できない個体として扱い，E_t に大きな値を与えた．

距離に関する評価は，各機械が割り当てられた圃場を順に作業する時の最短巡回経路を求めてその距離を用いるべきであるが，第 2 章で述べた様に n_f 個の圃場を割り当てられた時の巡回経路の組み合わせは $(n_f - 1)!/2$ 通り存在するため，n_f が大きくなると最短巡回経路を見つけるのが困難になる．これは第 2 章で述べた手法 [2] を用いて解を得ることができるが，個体の適応度は各機械につき $N_p \cdot N_g$(個体数×最終世代数) 回計算する必要がある．したがって，$N_p = 400, N_g = 400$ である時この計算は 160,000 回行なう必要があり，1 回の計算時間を 10 秒としても 400 時間かかるため現実的ではない．そこで距離に関する評価として式 (3.2) を用いて計算した．

$$E_d = \frac{2}{n_f(n_f - 1)} \sum_{i=1}^{n_f-1} \sum_{j=i+1}^{n_f} d_{ij} \tag{3.2}$$

ここで

E_d : 距離に関する評価

n_f : 割り当てられた圃場数

d_{ij} : 割り当てられた圃場のうち i 番目と j 番目の圃場間の距離

であり，E_d は割り当てられた全圃場のうちの二圃場間の距離の全ての値の平均値となる．なお，$n_f < 2$ の時，すなわち割り当てられた圃場数が 1，もしくは全く割り当てられなかった時は，その個体を環境に適応できない個体として扱い，E_d に大きな値を与えた．

これらの式 (3.1)，および式 (3.2) による値を用いて式 (3.3) により個体の評価 E を求めた．式 (3.3) で個体の評価は距離に関する評価をもとにしており，これに時間に関する評価によるペナルティを掛けて求めている．ペナルティは時間に関する評価に応じて変化するものとし，式 (3.3) に示すように一次式 $(aE_t + 1)$ を用いた．これにより，個体の評価は $E_d \leq E \leq (a+1)E_d$ となり，最大で $(a+1)$ 倍のペナルティを受けることとなる．適応度 F は式 (3.4) により個体の評価 E の逆数として求めたので，E の値が小さいほど適応度の高い個体となる．

$$E = (aE_t + 1)E_d \tag{3.3}$$

$$F = \frac{1}{E} \tag{3.4}$$

ここで

E : 個体の評価

a : 係数

F : 個体の適応度

である.

3.2.5 計算条件

計算に用いた圃場データは第2章と同様に，京都府船井郡和知町の財団法人「ふるさと振興センター」の農作業受託部が作業を受託した圃場のものである．これらの圃場の面積，各圃場間の距離，および機械の圃場作業量をデータとして与え，田植え作業を想定して計算を行なった．所有機械として圃場作業量の異なる3台の機械を想定し，実際の作業日誌のデータの平均値よりそれぞれの一日の圃場作業量を0.72，0.80，0.96ha [3]，一日の作業時間を8:30-18:30(昼休み1時間，休憩1時間)の8.0hとした．この一日の作業時間に実作業率0.625を掛け，一日の実作業時間を5.0h [4]，それぞれの圃場作業量(field capacity)を0.144，0.160，0.192ha/hとした．計算は表3.1に示すように突然変異率P_mを4段階，係数aを5段階に変化させ，それぞれについて20回行い，得られた解を比較した．なお，交叉数については，第2章の結果よりそれを変化させても得られた解に大きな差がなかったためここでは固定した．

表3.1: パラメータ
Table 3.1: Parameters

最終世代数:N_g	400
個　体　数:N_p	400
交　叉　数:N_c	160
突然変異率:P_m(%)	1.0, 4.0, 7.0, 10.0
係　　数:a	0.5, 1.0, 1.5, 2.0, 2.5

3.2.6 計算結果

計算に用いた全圃場の総面積は 370.9a で，全ての作業を最短作業期間で終了するための機械の必要最少台数は，総面積，各機械の圃場作業量，および一日の実作業時間より 5 台である．係数 a を変化させることで適応度における E_t の影響が変化し，a が大きくなると一日の実作業時間との誤差が小さくなるように，a が小さくなると圃場間の移動距離が短くなるように圃場が割り当てられる．表 3.2 は各パラメータの組み合わせで 20 回ずつ計算した時の圃場を割り当てられた機械数の平均値，表 3.3 は最小値である．表 3.2 が示

表 3.2: 機械数の平均値
Table 3.2: Mean of number of machines

		Mutation probability (%)			
		1.0	4.0	7.0	10.0
Coefficient 'a'	0.5	7.60	10.85	10.50	8.95
	1.0	6.05	7.70	8.00	8.00
	1.5	5.70	6.25	6.35	6.80
	2.0	5.30	5.90	5.95	6.15
	2.5	5.35	5.90	5.75	5.80

表 3.3: 機械数の最小値
Table 3.3: Minimum of number of machines

		Mutation probability (%)			
		1.0	4.0	7.0	10.0
Coefficient 'a'	0.5	6	6	6	6
	1.0	5	6	6	6
	1.5	5	5	5	6
	2.0	5	5	5	5
	2.5	5	5	5	5

すように，20 回行なった計算結果がすべて必要最少台数である 5 台となるパラメータの組み合わせはなかった．また，表 3.3 が示すように，係数 a が 0.5 の時は突然変異率にかかわらず，必要最少台数である 5 台となる解は得られなかった．これは E_t よりも E_d の影響が大きくなり，作業時間の誤差が大きくても移動距離が短くなるような圃場割り当てを示す個体の適応度が高くなるためである．その結果，各機械に割り当てられる圃場は少な

くなり，必要な機械数が増えたものと考えられる．

この手法では各機械について順番に圃場を割り当てるため，n_m 台目の機械への圃場割り当ての計算は $n_m - 1$ 台目までの機械への圃場割り当ての計算結果により初期条件が変化し影響を受けるが，逆に $n_m - 1$ 台目までの計算に n_m 台目の計算結果が影響を与えることはない．したがって，n_m が大きくなるにつれて，すなわち後から圃場を割り当てられる機械ほど圃場の位置の初期条件が悪くなる．表3.4はこの手法での計算結果の一例である．この結果は係数 a が2.0，突然変異率が10.0%の時に得られたものである．図3.4と表3.4により機械1では地図中の全ての圃場から作業をする圃場を割り当てられるため，狭い範囲に存在する圃場が割り当てられている様子が示されている．しかし，機械1に割り当てられた圃場は機械2以降には割り当てられないため，これらの圃場により1から26までの圃場と34から38の圃場が分断されてしまっている．これらのうち34から38の5つの圃場の面積の和は42.5aであり，圃場作業量の最も小さい機械でも時間に関する評価 E_t の値，すなわち所要時間と実作業時間との誤差の割合が0.41と大きくなっている．したがって，この誤差の割合を少なくするためには，他の圃場での作業も割り当てる必要があるが，27から33の圃場が機械1に割り当てられているため，これらの圃場から離れている1から26の圃場での作業が割り当てられることになる．その結果，距離に関する評価 E_d が大きくなる．圃場割り当てを進めるにつれてこのように分断された圃場が増え，機械4および5では割り当てられた圃場が点在している．特に機械5に割り当てられた圃場は，機械4までに割り当てられなかった残りの圃場であるため互いの距離が大きくなっている．

表 3.4: 計算結果 (手法 1) $((P_m, a) = (10.0, 2.0))$
Table 3.4: A result of calculation (method 1) $((P_m, a) = (10.0, 2.0))$

Machine No.	No. of field assigned	E_t	E_d
1	27, 28, 29, 30, 31, 32, 33	0.18	271.59
2	5, 6, 7, 8, 9, 10, 13	0.021	195.40
3	12, 14, 16, 18, 20, 21, 22	0.043	206.90
4	1, 2, 3, 4, 11, 15	0.16	314.48
5	17, 19, 23, 24, 25, 26, 34, 35, 36, 37, 38	0.024	1436.87

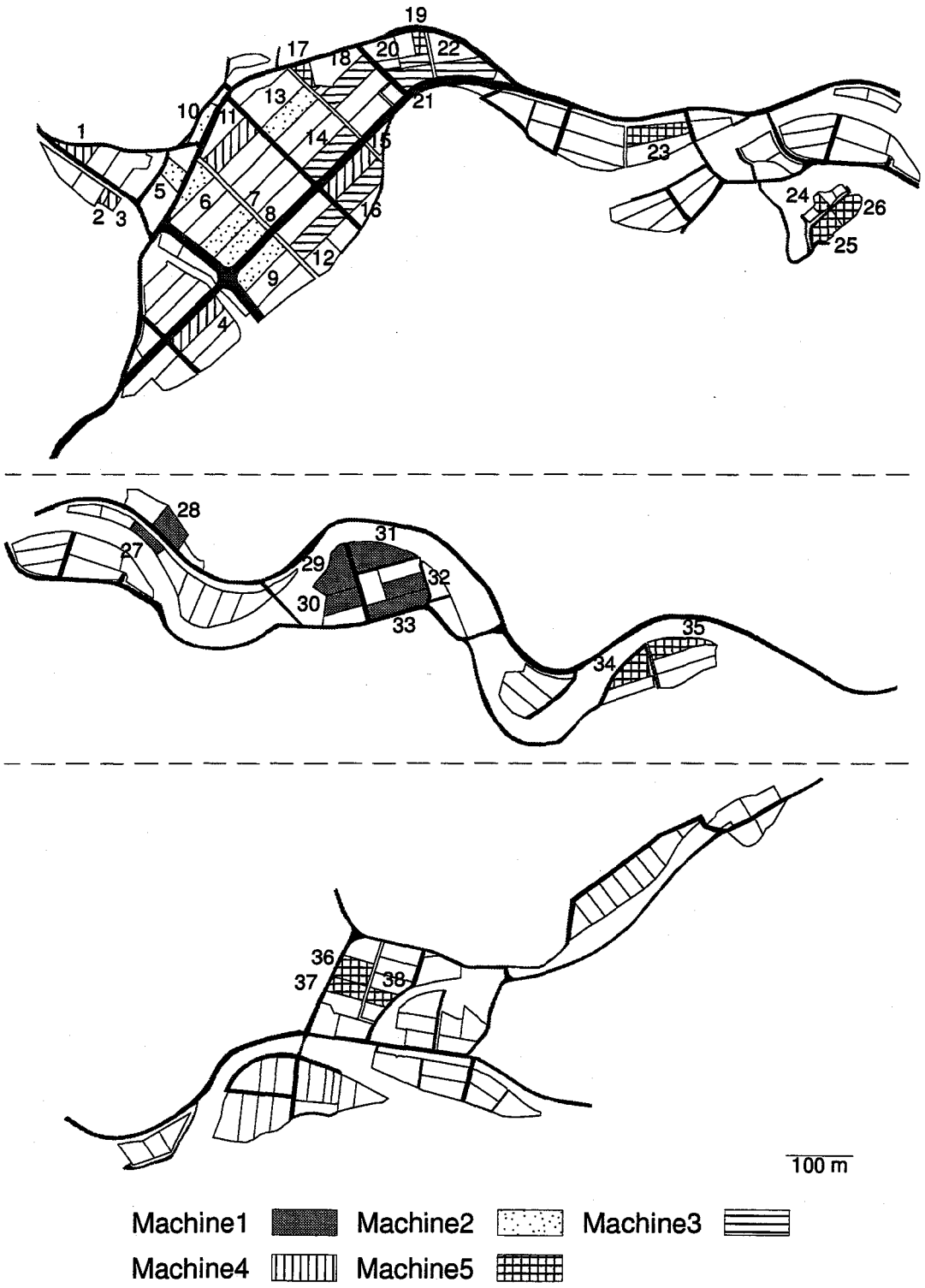


図 3.4: 5 台の田植機への圃場割り当て ($(P_m, a) = (10.0, 2.0)$)

Figure 3.4: Assignment of fields to 5 rice transplanters ($(P_m, a) = (10.0, 2.0)$)

これらの結果より，この手法では係数 a，突然変異率 P_m の組み合わせを変えても，常に最適解であると思われる必要な機械数が必要最少台数となる解を得るのは困難であると考えられる．

3.3 必要最少台数の機械に同時に割り当てる手法(手法2)

手法1で明らかになった問題点に対処するためにアルゴリズムを変更し，それに伴ってコード化，突然変異の方法，適応度の計算方法を変更した．

3.3.1 変更点

手法1で問題となったのは，後から圃場を割り当てられる機械での計算の初期条件が悪化することである．これは手法1が各機械に一台ずつ順番に作業を行なう圃場を割り当てたからである．そこで，手法2ではまず最初に全作業を終了するのに必要な機械数を求め，その必要最少台数分の機械に同時に圃場を割り当てるように変更した [5]．図3.5にこの手法で用いたコード化の方法で生成した個体の染色体を示す．手法1の個体と同様に遺伝子長は圃場数と一致し，各遺伝子座に圃場が対応している．各遺伝子は機械の番号に対応する数値をとり，その遺伝子座に対応する圃場で作業する機械を表す．図3.5は圃場数が10の時の個体の染色体で，2, 3, 5, 9の各圃場を機械1で，1, 6, 10を機械2で，4, 7, 8を機械3で作業することを表している．コード化の変更に伴い突然変異の方法も変更した．図3.6に示すように，ランダムに選ばれた遺伝子座の遺伝子を他の値に変化させることで突然変異を起こした．交叉は手法1と同様に一点交叉を採用した．

2	1	1	3	1	2	3	3	1	2

図 3.5: 手法2の染色体
Figure 3.5: A chromosome of the method 2

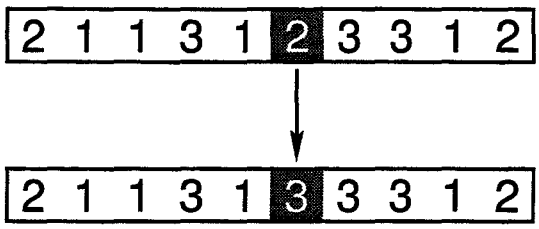


図 3.6: 手法 2 の突然変異
Figure 3.6: The strategy of mutation for method 2

3.3.2 適応度

この手法では手法 1 と同様に式 (3.1), 式 (3.2) により各機械についての時間および距離に関する評価を求め, 式 (3.3) により各機械の評価を計算し, これらの和により個体の評価を求めた (式 (3.5)). 個体の適応度は手法 1 と同様に個体の評価の逆数として式 (3.6) により求めた.

$$E = \sum_{i=1}^{N_m} E_i \tag{3.5}$$

$$F = \frac{1}{E} \tag{3.6}$$

ここで

E : 個体の評価

E_i : 機械 i の評価

N_m : 機械の必要最少台数

F : 個体の適応度

である.

3.3.3 計算条件

面積, 圃場間の距離などの圃場データ, 機械の圃場作業量, および一日の実作業時間は手法 1 と同様のものを使い, 圃場作業量の異なる 3 台の所有機械を用いた田植え作業を想定して計算を行なった. 計算に用いたパラメータを表 3.5 に示す. 変化させたパラメータは係数 a および突然変異率 P_m で, a は 5 段階に, P_m は 6 段階に変化させ, 各計算条件につき 50 回ずつ計算し, 結果を比較した.

表 3.5: パラメータ
Table 3.5: Parameters

最終世代数:N_g	400
個 体 数:N_p	400
交 叉 数:N_c	160
突然変異率:P_m(%)	1.0, 4.0, 7.0, 10.0, 13.0, 16.0
係 数:a	0.5, 1.0, 1.5, 2.0, 2.5

3.3.4 計算結果

この手法では最初に全圃場での作業に必要な機械の数を計算し，それらの機械に作業を行なう圃場を同時に割り当てているため，すべての解で5台の機械に圃場が割り当てられている．なお，5台目の機械の一日の実作業時間Tについては，4台目までの機械が5時間ずつ作業した残りの圃場で作業するのに必要な時間を計算し，3.18hとした．

図3.7に突然変異率と50回の計算での適応度の平均値および95%信頼区間の関係を示す．係数aの値に関わらず突然変異率10.0%および7.0%の時適応度が高くなっているのが示されている．これにより，この手法では突然変異率を10.0%もしくは7.0%に設定するのが適していることが明らかになった．

これらの突然変異率で計算した時の結果のうち，各パラメータの組み合わせで最も適応度の高かったものを図3.8〜図3.13に，各機械の時間に関する評価および距離に関する評価を表3.6〜表3.11に示す．図3.8，表3.6は$(P_m, a) = (7.0, 0.5), (10.0, 0.5)$，図3.9，表3.7は$(P_m, a) = (7.0, 1.0)$，図3.10，表3.8は$(P_m, a) = (7.0, 1.5)$，図3.11，表3.9は$(P_m, a) = (7.0, 2.0), (7.0, 2.5), (10.0, 2.0), (10.0, 2.5)$，図3.12，表3.10は$(P_m, a) = (10.0, 1.0)$，図3.13，表3.11は$(P_m, a) = (10.0, 1.5)$の時の結果である．図3.8，表3.6，および図3.11，表3.9でパラメータの組み合わせが複数あるのはそれぞれのパラメータで同じ結果が得られたためである．図3.9と図3.10，および図3.11と図3.13はそれぞれ機械1と機械4に割り当てられた圃場が入れ替わったものである．機械4は機械1の二日目の作業を表すため，適応度の計算に作業日を考慮に入れていない今回の手法ではこれらの解を表す個体の適応度は等しくなり，どちらがより適した作業計画を表しているかを判断することはできない．また，図3.7では突然変異率が同じである時，aの値が小さいほど適応度が高くなっているように見えるが，aを変化させると評価関数が変化するため，aが異なるときの適応度

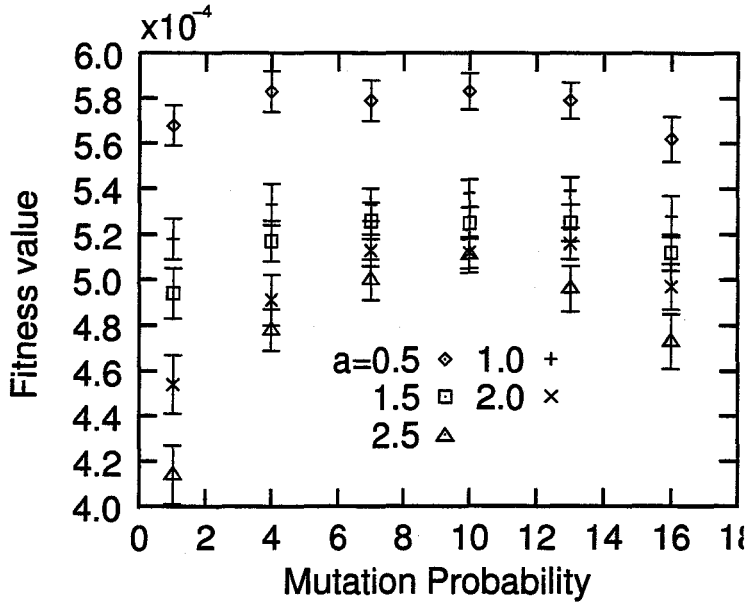


図 3.7: 突然変異率と適応度の関係

Figure 3.7: The relations between mutation probability and the fitness value

を比較することには意味がない．つまり a が異なる，すなわち評価関数が異なるということは，それぞれの個体が属する環境が異なるということであり，それぞれの環境に対する適応度は個体の絶対的な優劣を示すものではない．

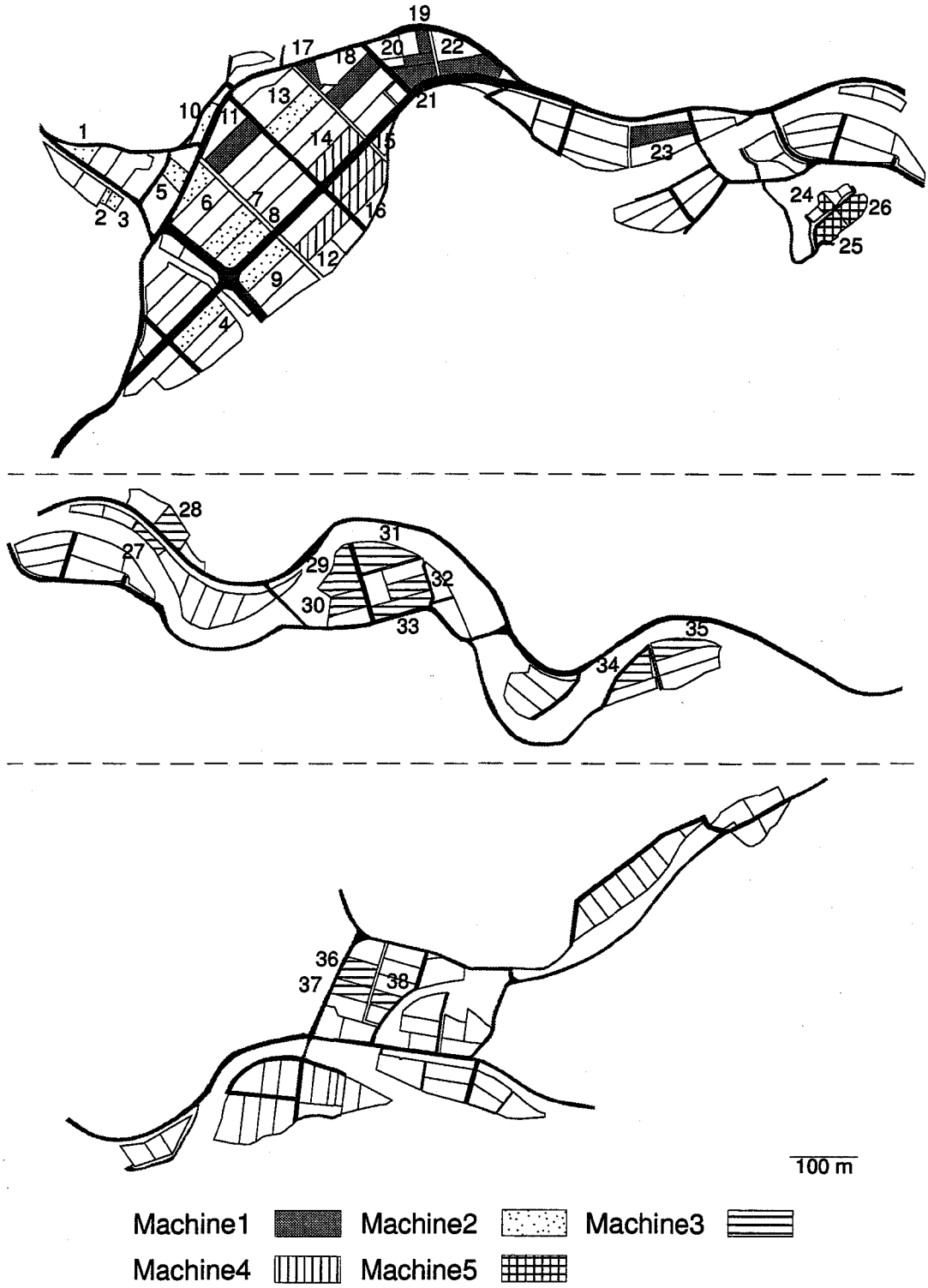


図 3.8: 5台の田植機への圃場割り当て ((P_m, a) = (7.0, 0.5), (10.0, 0.5))

Figure 3.8: Assignment of fields to 5 rice transplanters ((P_m, a) = (7.0, 0.5), (10.0, 0.5))

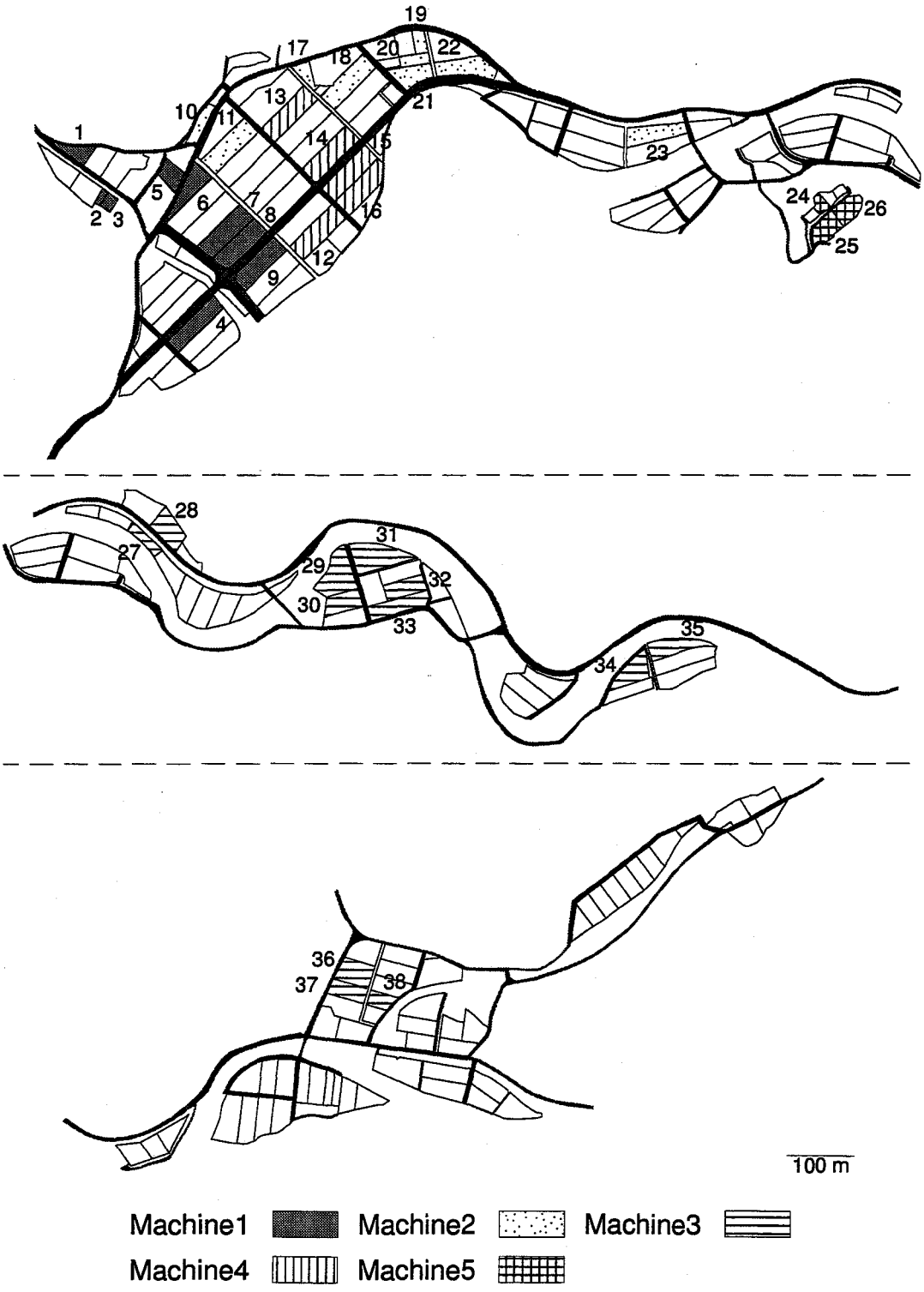


図 3.9: 5 台の田植機への圃場割り当て ((P_m, a) = (7.0, 1.0))

Figure 3.9: Assignment of fields to 5 rice transplanters ((P_m, a) = (7.0, 1.0))

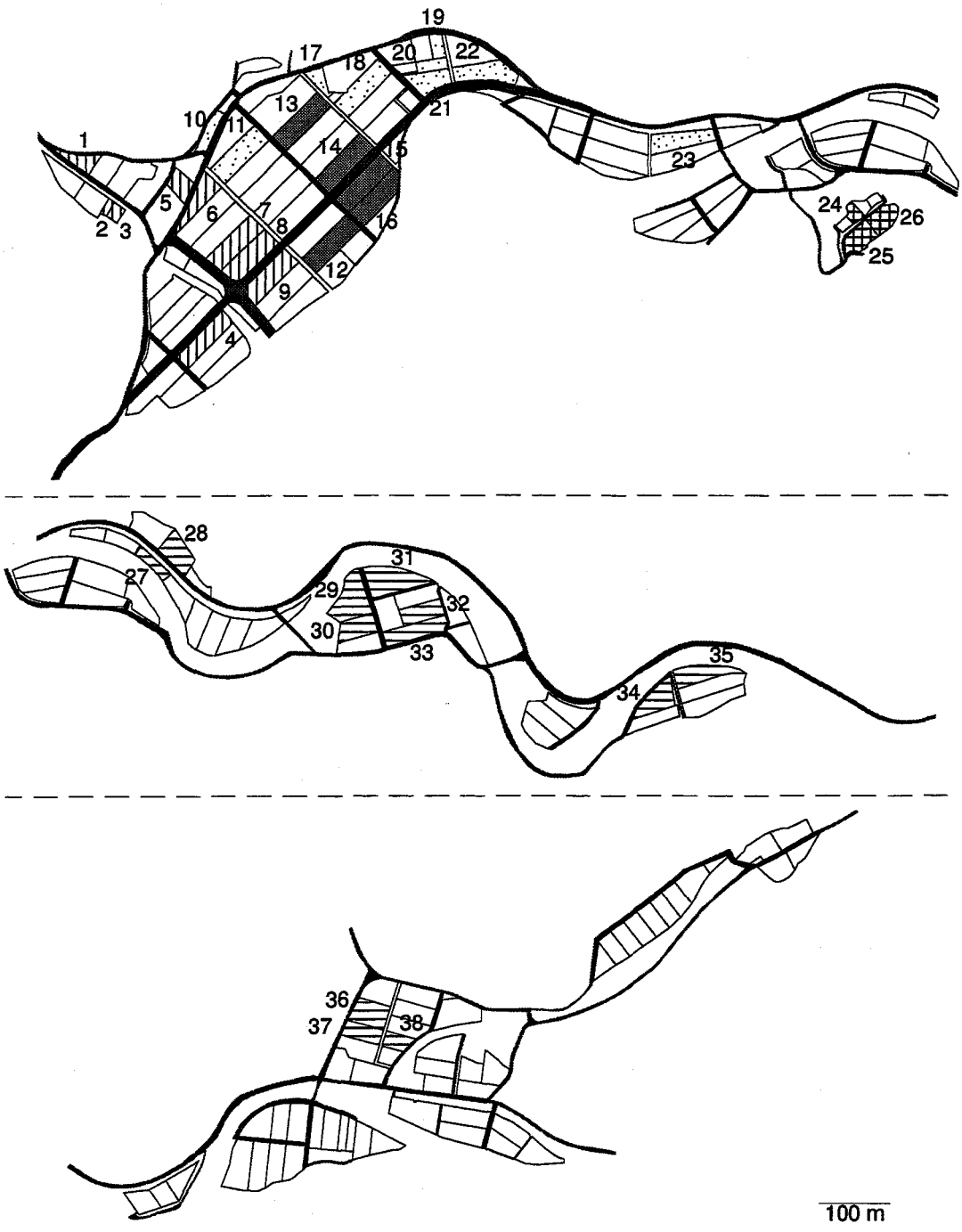


図 3.10: 5台の田植機への圃場割り当て ($(P_m, a) = (7.0, 1.5)$)

Figure 3.10: Assignment of fields to 5 rice transplanters ($(P_m, a) = (7.0, 1.5)$)

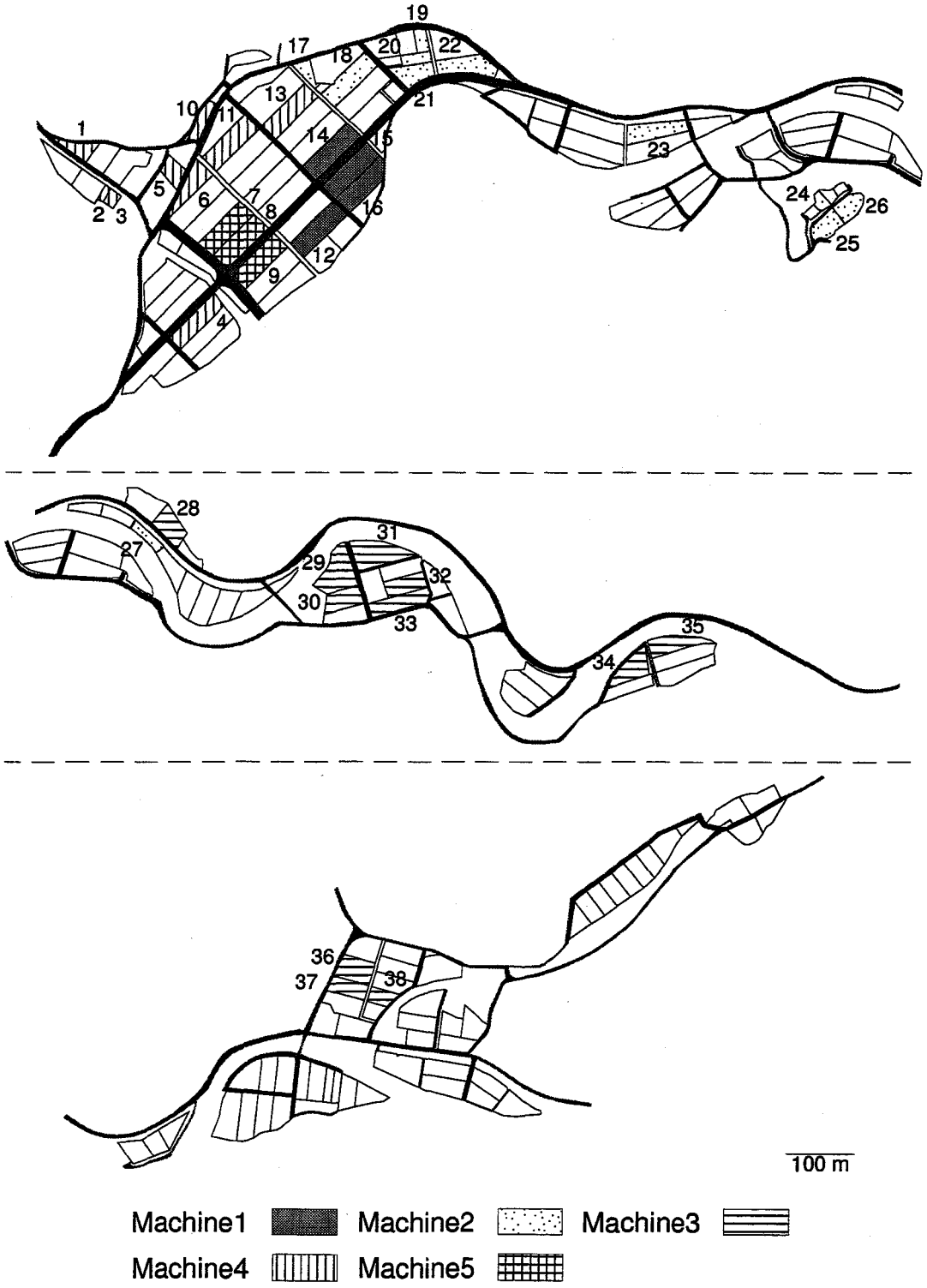


図 3.11: 5 台の田植機への圃場割り当て ((P_m, a) = (7.0, 2.0), (7.0, 2.5), (10.0, 2.0), (10.0, 2.5))

Figure 3.11: Assignment of fields to 5 rice transplanters ((P_m, a) = (7.0, 2.0), (7.0, 2.5), (10.0, 2.0), (10.0, 2.5))

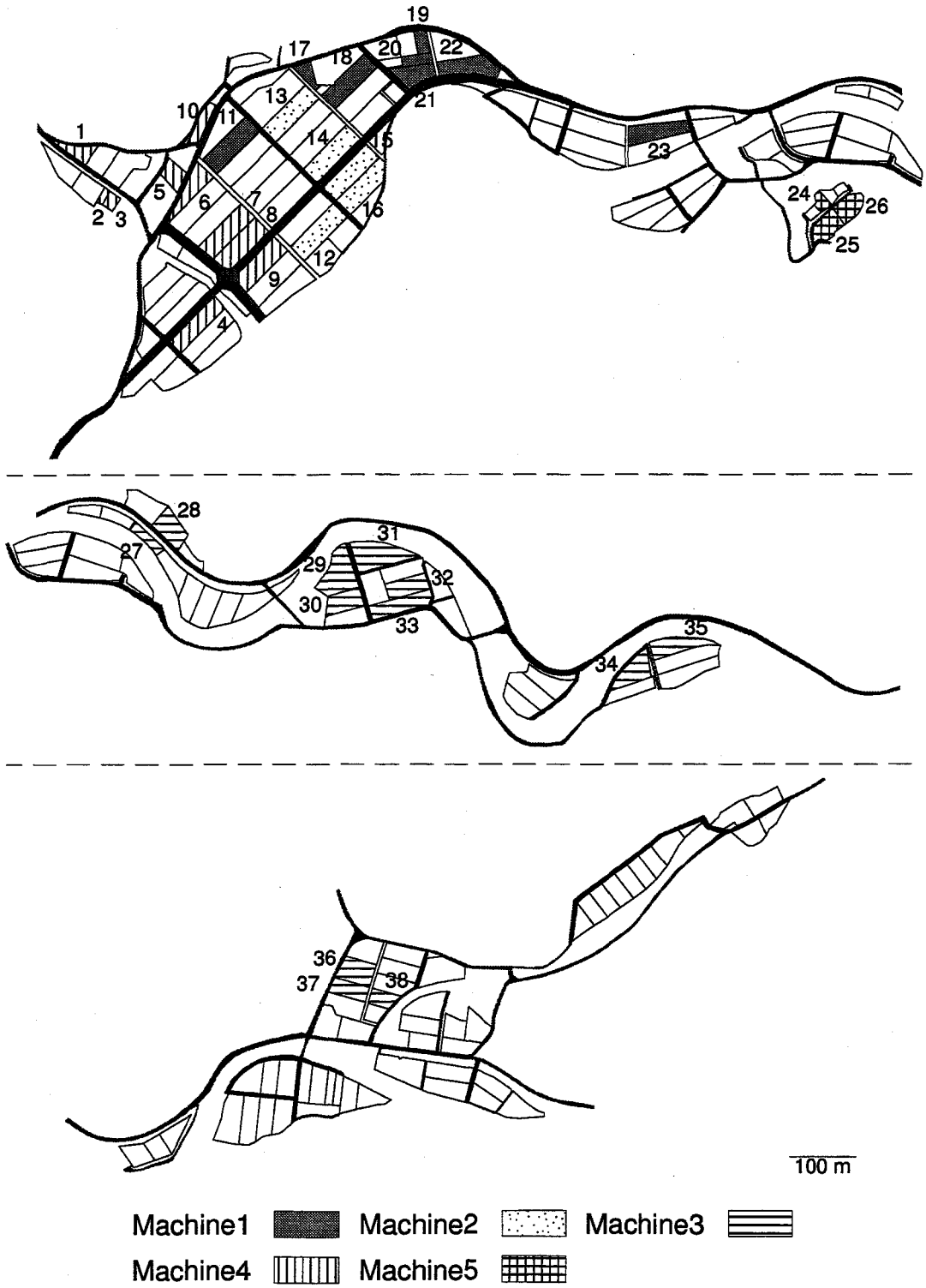


図 3.12: 5 台の田植機への圃場割り当て ((P_m, a) = (10.0, 1.0))

Figure 3.12: Assignment of fields to 5 rice transplanters ((P_m, a) = (10.0, 1.0))

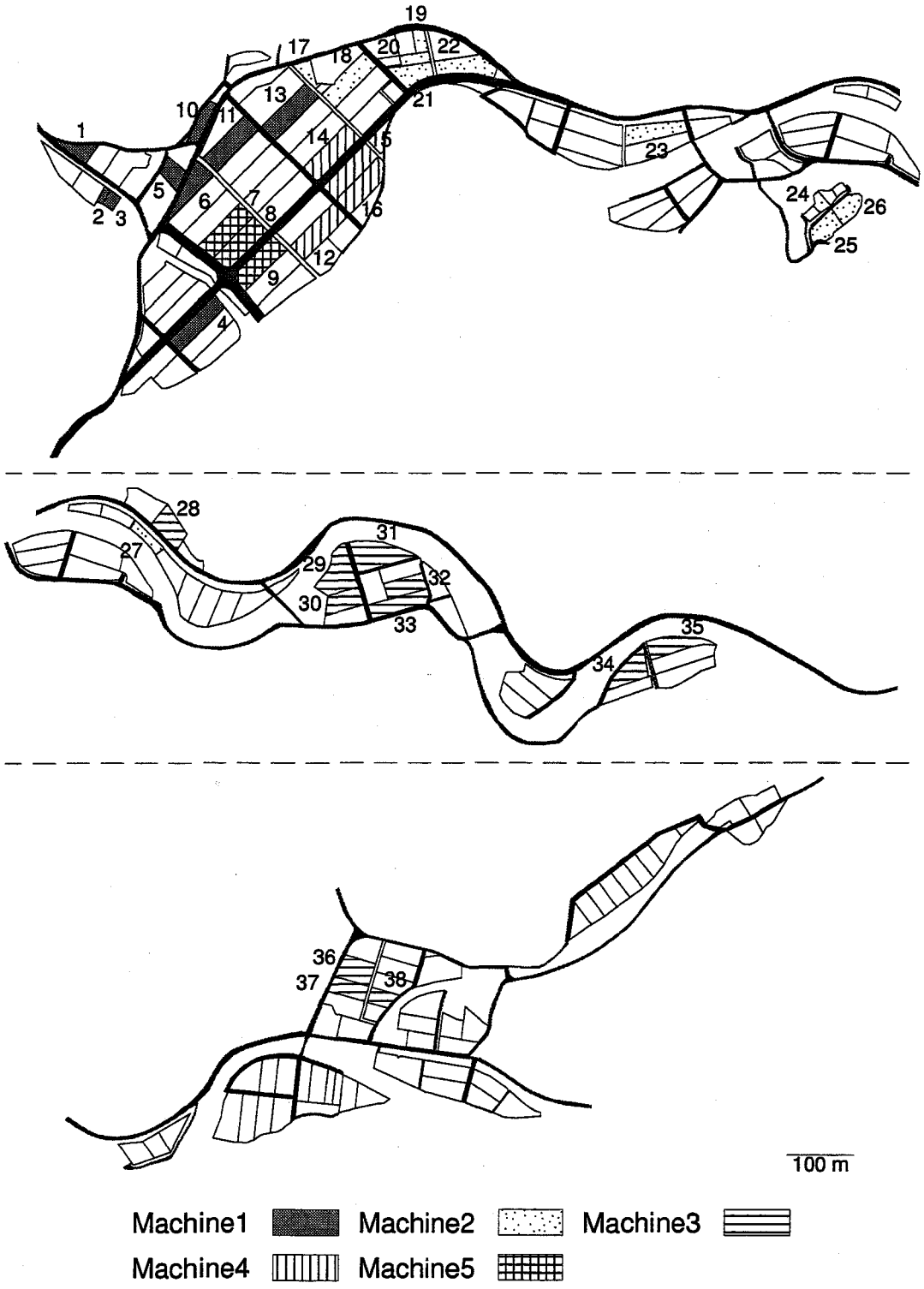


図 3.13: 5 台の田植機への圃場割り当て ($(P_m, a) = (10.0, 1.5)$)

Figure 3.13: Assignment of fields to 5 rice transplanters ($(P_m, a) = (10.0, 1.5)$)

表 3.6: 計算結果 $((P_m, a) = (7.0, 0.5), (10.0, 0.5))$

Table 3.6: The result of calculation $((P_m, a) = (7.0, 0.5), (10.0, 0.5))$

Machine No.	No. of field assigned	E_t	E_d
1	11, 17, 18, 19, 20, 21, 22, 23	0.090	288.92
2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13	0.35	235.99
3	27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38	0.057	864.21
4	12, 14, 15, 16	0.039	48.28
5	24, 25, 26	0.73	50.57

表 3.7: 計算結果 $((P_m, a) = (7.0, 1.0))$

Table 3.7: The result of calculation $((P_m, a) = (7.0, 1.0))$

Machine No.	No. of field assigned	E_t	E_d
1	1, 2, 3, 4, 5, 6, 7, 8, 9	0.20	210.44
2	10, 11, 17, 18, 19, 20, 21, 22, 23	0.033	300.86
3	27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38	0.057	864.21
4	12, 13, 14, 15, 16	0.20	91.72
5	24, 25, 26	0.73	50.57

表 3.8: 計算結果 $((P_m, a) = (7.0, 1.5))$

Table 3.8: The result of calculation $((P_m, a) = (7.0, 1.5))$

Machine No.	No. of field assigned	E_t	E_d
1	12, 13, 14, 15, 16	0.20	91.72
2	10, 11, 17, 18, 19, 20, 21, 22, 23	0.033	300.86
3	27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38	0.057	864.21
4	1, 2, 3, 4, 5, 6, 7, 8, 9	0.20	210.44
5	24, 25, 26	0.73	50.57

表 3.9: 計算結果 $((P_m, a) = (7.0, 2.0), (7.0, 2.5), (10.0, 2.0), (10.0, 2.5))$
Table 3.9: The result of calculation $((P_m, a) = (7.0, 2.0), (7.0, 2.5), (10.0, 2.0), (10.0, 2.5))$

Machine No.	No. of field assigned	E_t	E_d
1	12, 14, 15, 16	0.039	48.28
2	17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27	0.010	548.65
3	28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38	0.013	858.37
4	1, 2, 3, 4, 5, 6, 10, 11, 13	0.042	241.57
5	7, 8, 9	0.012	52.87

表 3.10: 計算結果 $((P_m, a) = (10.0, 1.0))$
Table 3.10: The result of calculation $((P_m, a) = (10.0, 1.0))$

Machine No.	No. of field assigned	E_t	E_d
1	11, 17, 18, 19, 20, 21, 22, 23	0.090	288.92
2	12, 13, 14, 15, 16	0.084	91.71
3	27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38	0.057	864.21
4	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	0.26	219.39
5	24, 25, 26	0.73	50.57

表 3.11: 計算結果 $((P_m, a) = (10.0, 1.5))$
Table 3.11: The result of calculation $((P_m, a) = (10.0, 1.5))$

Machine No.	No. of field assigned	E_t	E_d
1	1, 2, 3, 4, 5, 6, 10, 11, 13	0.042	241.57
2	17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27	0.010	548.65
3	28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38	0.013	858.37
4	12, 14, 15, 16	0.039	48.28
5	7, 8, 9	0.012	52.87

これらの表が示すように，aの値が小さいと距離に関する評価E_dを重視した作業計画が得られ，全機械の総移動距離は小さくなる一方，時間に関する評価E_tの値が大きくなり各機械の作業時間が均等ではなくなっている．逆に，aが大きいと総移動距離は増加するが，各機械の作業時間はほぼ均等になる．このようにaを変化させることで得られる解にどのような影響を与えるかを調べるためには，時間に関する評価E_t，および距離に関する評価E_dを比較する必要がある．そこで，式(3.7)，式(3.8)により解として得られた個体について，時間に関する評価の和S_t，距離に関する評価の和S_dを求めて比較した．

$$S_t = \sum_{i=1}^{N_m} E_{ti} \tag{3.7}$$

$$S_d = \sum_{i=1}^{N_m} E_{di} \tag{3.8}$$

ここで

E_{ti}：機械iの時間に関する評価

E_{di}：機械iの距離に関する評価

である．

図3.14に突然変異率を10.0%に設定して50回ずつ計算した時の，aとS_tの平均値および95%信頼区間の関係を，図3.15にaとS_dの平均値および95%信頼区間の関係を示す．これらの図が示すように，aの値を大きくすると時間に関する評価の和S_tは小さくなり，全機械の総移動距離S_dは大きくなる．これらのことからaを1.5もしくは2.0に設定することで各機械の作業時間の誤差率，すなわち時間に関する評価E_tが約0.12(約30分)と均等になり，総移動距離が少なくなるように圃場を割り当てることができるものと考えられる．aを2.0に設定して得られた解で最も適応度が高かったのは図3.11，表3.9で示したものであるが，これらの図と表から各機械とも狭い範囲に存在する圃場での作業を割り当てられており，圃場間の移動距離が短い結果となっていることが示されている．また所要時間と実作業時間の誤差率も最大で0.042と少なくなっており，これは各機械に均等に割り当てられていることを示している．

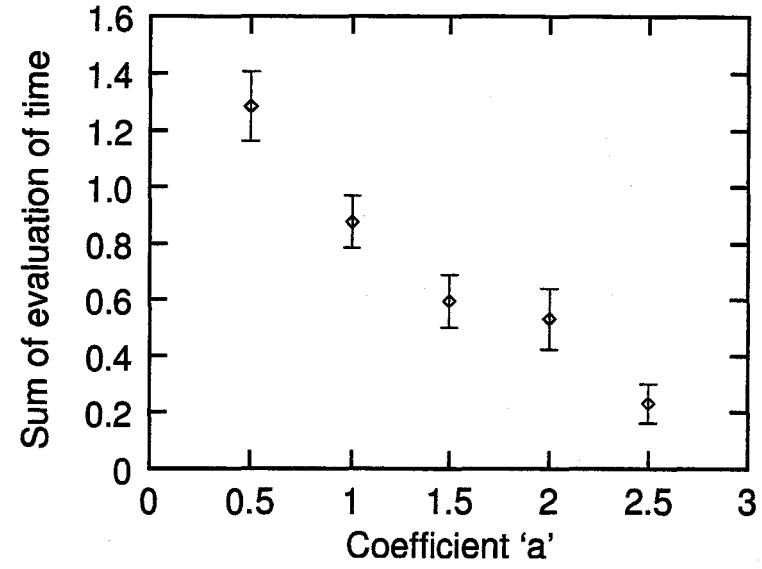


図 3.14: 係数 a と時間による評価の関係

Figure 3.14: The relation between the coefficient 'a' and the evaluation of time

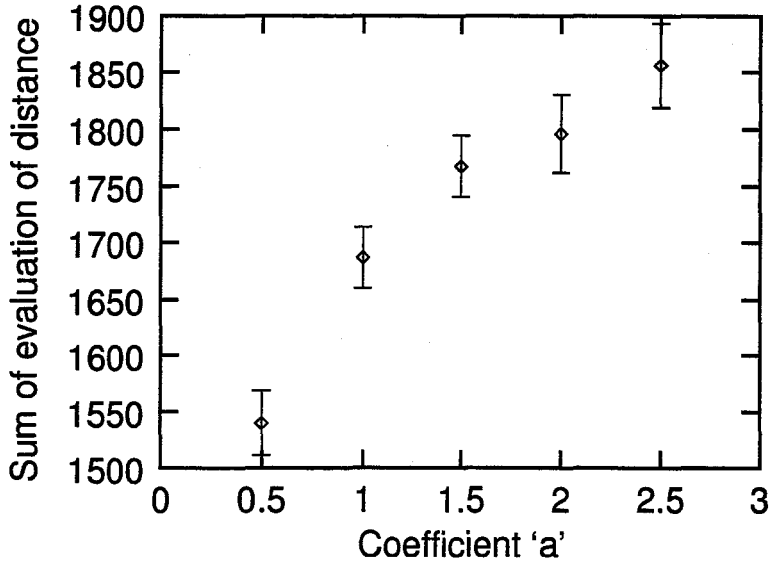


図 3.15: 係数 a と距離による評価の関係

Figure 3.15: The relation between the coefficient 'a' and the evaluation of distance

3.4 ランク選択を用いた手法との比較

第2章では選択の方法として，適応度に比例して選択されるルーレット選択を用いた手法と，集団の中での順位によって選択されるランク選択を用いた手法で計算した．それぞれの手法で得られた解の平均値を比較したところ，明らかにランク選択を用いた手法の方が良い結果となっていた．そこで，本章で用いた手法においてもこのランク選択を採用し，さらに良好な解が得られるかを検討した．

3.4.1 計算条件

ルーレット選択を用いた時の結果と比較するため，面積，圃場間の距離などの圃場データ，機械の圃場作業量，および一日の実作業時間は同様のものを使い，圃場作業量の異なる3台の所有機械を用いた田植え作業を想定して計算を行なった．計算に用いたパラメータを表3.12に示す．変化させたパラメータは係数aおよび突然変異率P_mで，aは5段階に，P_mはルーレット選択を用いた時に適応度の高い個体を結果として得ることができた7.0%, 10.0%の2段階に変化させ，各計算条件につき50回ずつ計算し，ルーレット選択を用いた時の結果と比較した．

表3.12: パラメータ
Table 3.12: Parameters

最終世代数:N_g	400
個　体　数:N_p	400
交　叉　数:N_c	160
突然変異率:P_m(%)	7.0, 10.0
係　　数:a	0.5, 1.0, 1.5, 2.0, 2.5

3.4.2 計算結果

図3.16に$P_m = 7.0\%$，図3.17に$P_m = 10.0\%$の時のaと適応度の関係を示す．これらの図よりaが0.5および1.0ではルーレット選択とランク選択で得られる解にほとんど差がなく，1.5以上では大きな差ではないがルーレット選択の方が得られた解の適応度の平均

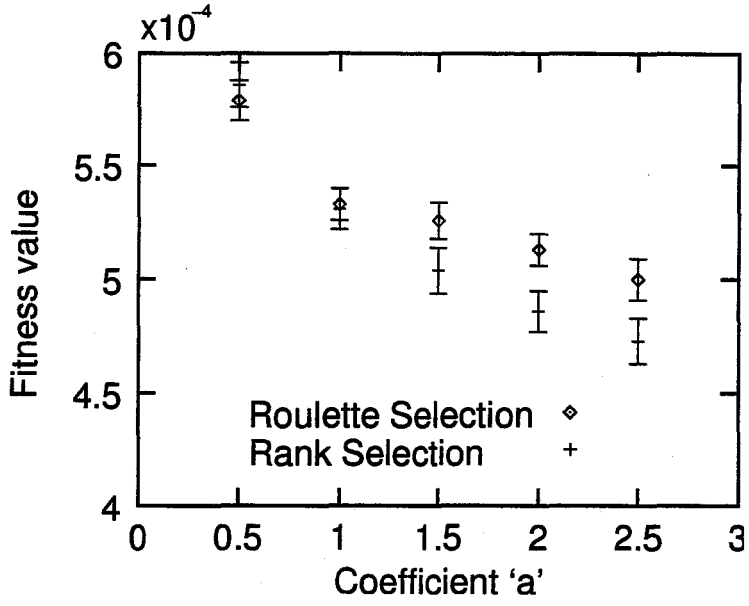


図 3.16: 係数 a と適応度の関係 (P_m = 7.0%)

Figure 3.16: The relations between the coefficient 'a' and the fitness value (P_m = 7.0%)

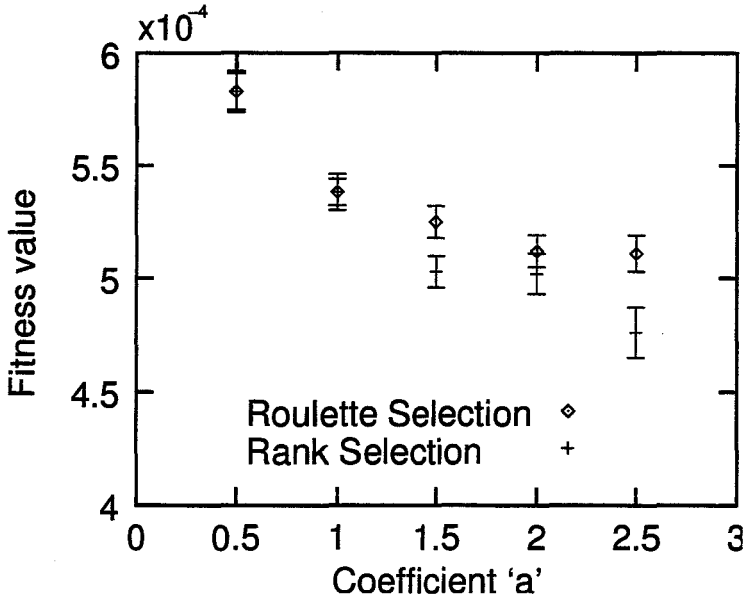


図 3.17: 係数 a と適応度の関係 (P_m = 10.0%)

Figure 3.17: The relations between the coefficient 'a' and the fitness value (P_m = 10.0%)

値が高くなっているが，第2章の結果の様な明らかな差ではなかった．そこで，これらの差が有意であるかを検討するために解の集団を統計的に処理した．

平均値の差の有意性を検定するにはt検定を行なうのが一般的であるが，このt検定では母集団が正規分布に従っていることを仮定している．GAで得た解の適応度の値を要素とする集団の分布は，パラメータの組み合わせが理想的な時には最適解となる適応度の値が最頻値となり適応度の値が下がるに従って頻度が下がる．この時，最適解となる適応度の値より大きな値が出現することはないため，正規分布の片側の様な分布となる．これに対して，パラメータの組み合わせが適切でない時には正規分布の様な分布となる．図3.18はパラメータの組み合わせが適切な時 $((P_m, a) = (10.0, 1.5))$ の，図3.19は適切でない時 $((P_m, a) = (1.0, 2.5))$ の解の適応度の値の分布を示したものである．これらの図が示すようにパラメータの組み合わせが適切な時には最頻値は最適解の付近にあり，適切でない時は最頻値は分布のほぼ中央に現れる．これらのことからGAで得た解の適応度は正規分布に従わないと考えられるため，t検定を用いることはできない．そこでこれらの集団の分布の代表値の差についてMann-WhitneyのU検定 [6] を用いて有意性を検定した．U検定はケース数が n_1, n_2 である2つの群の観察値を小さい順に並べて，群ごとの順位の和を検定統計量にするものである．両選択方法で同一パラメータで50回ずつ計算した解の適応度の値について検定統計量 U を求め，式(3.9)により Z_0 を計算した．この Z_0 をもとに有意確率 $P = Pr\{|Z| \geq Z_0\}$ を正規分布表より求めた．これらの値を表3.13に示す．

$$Z_0 = \frac{|U - E(U)|}{\sqrt{V(U)}} \tag{3.9}$$

ここで

$E(U)$: 検定統計量 U の平均値

$V(U)$: 検定統計量 U の分散

であり，Z_0 は正規分布に従う．

表3.13より有意水準5%で $(P_m, a) = (7.0, 0.5), (7.0, 1.0), (10.0, 0.5), (10.0, 1.0), (10.0, 2.0)$ の時，両選択方法により得られる解の適応度の値の代表値には差がなく，$(P_m, a) = (7.0, 1.5), (7.0, 2.0), (7.0, 2.5), (10.0, 1.5), (10.0, 2.5)$ の時，両者に差がありルーレット選択を用いた方が良好な結果が得られることが判明した．

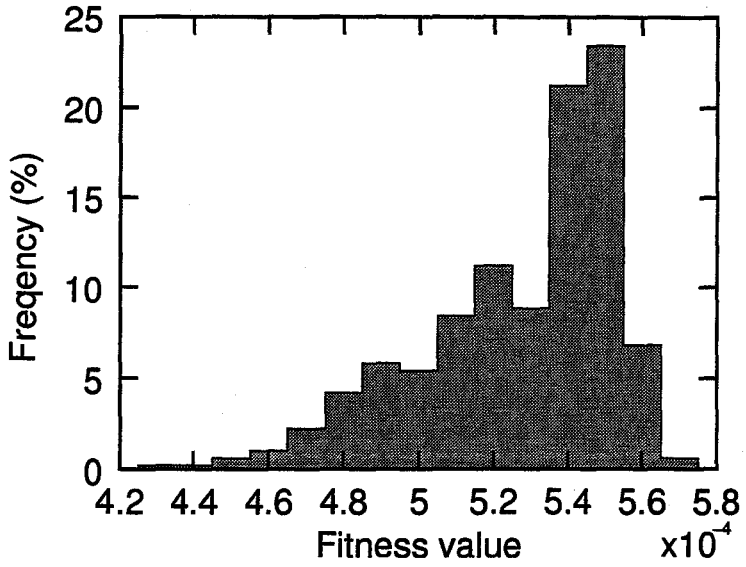


図 3.18: 解の適応度の分布 $((P_m, a) = (10.0, 1.5))$

Figure 3.18: The distribution of fitness value of result $((P_m, a) = (10.0, 1.5))$

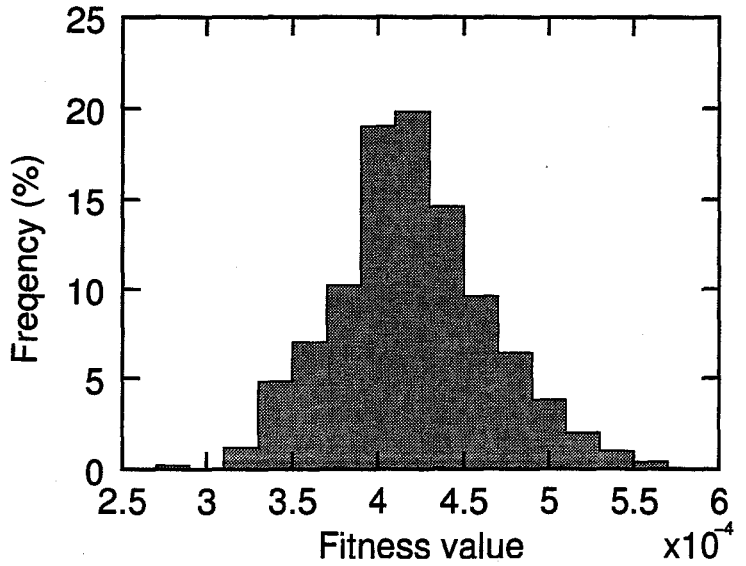


図 3.19: 解の適応度の分布 $((P_m, a) = (1.0, 2.5))$

Figure 3.19: The distribution of fitness value of result $((P_m, a) = (1.0, 2.5))$

表 3.13: 代表値の差の検定結果
Table 3.13: The results of test for difference of the average

P_m	a	U	Z_0	P
7.0	0.5	1163.0	0.60	5.485×10^{-1}
7.0	1.0	1169.0	0.56	5.755×10^{-1}
7.0	1.5	791.0	3.17	1.524×10^{-3}
7.0	2.0	645.5	4.17	3.046×10^{-5}
7.0	2.5	732.5	3.57	3.570×10^{-4}
10.0	0.5	1203.0	0.32	7.490×10^{-1}
10.0	1.0	1243.0	0.05	9.601×10^{-1}
10.0	1.5	663.5	4.04	5.345×10^{-5}
10.0	2.0	1009.5	1.66	9.691×10^{-2}
10.0	2.5	566.5	4.71	2.477×10^{-6}

3.5 考察

手法1では割り当てられた圃場数が少ない時に，所要時間と実作業時間の誤差が大きいにも関わらず，個体の適応度としては高くなることがあった．図3.20で左側は圃場数が6の時，右側は3の時の距離に関する評価のモデルを表している．この図で実線で示されたものは最短巡回経路の一部となる経路，破線で示されたものはそれ以外の経路である．したがって，破線で表した各圃場間の距離の平均値は実線で表した各圃場間の距離の平均値より大きくなる．距離に関する評価は，全ての圃場間の距離の平均値であるため，破線部が多くなると大きくなる．この破線部の数は割り当てられた圃場数が n_f の時 $n_f(n_f-3)/2$ となるため，n_f が大きくなるほど破線部の影響を受け，距離に関する評価の値は大きくなる．また $n_f=3$ の時には図でも明らかなように最短経路の平均値と一致する．よって，時間に関する評価が同じであれば，割り当てられた圃場数が少ない方が個体の適応度としては高くなる傾向がある．その結果，各機械に割り当てられた圃場での作業に必要な時間は一日の実作業時間より少なくなり，全作業を終了するためには必要最少台数を超える機械が必要となるケースが多く見られた．また，手法1では n_m 台目の機械への圃場割り当てが終了した時に，残りの圃場を n_m+1 台目の機械で一日で作業できるならば，それらの圃場を n_m+1 台目の機械に割り当てて計算を終了する．そのため，最後に割り当てられた機械の移動距離は長くなる傾向が強い．表3.4に示すように，5台目の機械では距離

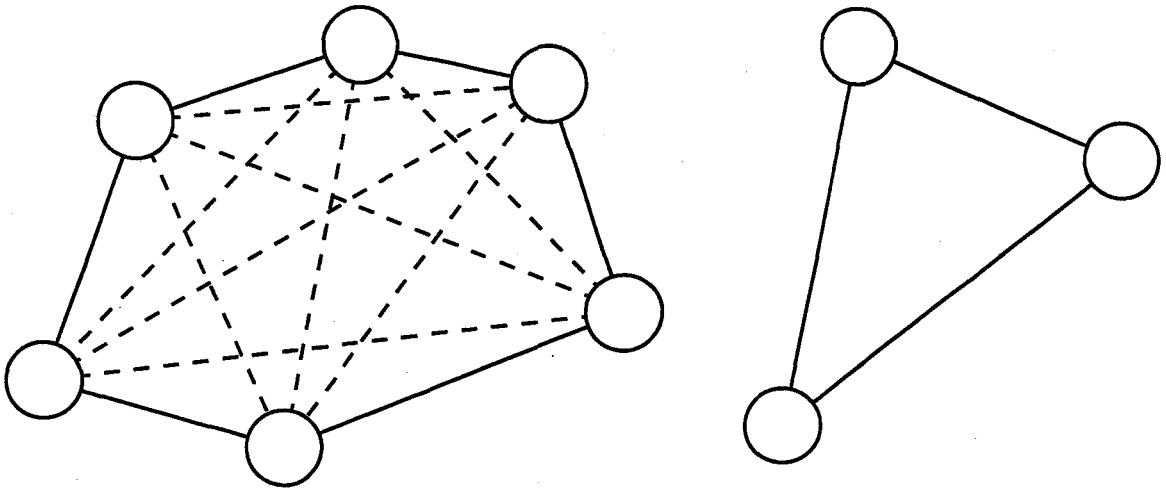


図 3.20: 距離に関する評価のモデル
Figure 3.20: The model of the evaluation of distance

に関する評価が著しく大きな値となっている.

手法2では手法1と同様の計算方法で各機械についての評価を求め，それらの和の逆数を個体の適応度としているため，各機械についての評価は手法1と同様に割り当てられた圃場数が少ない時に小さな値になるという傾向を示す．しかし，手法2ではあらかじめ全作業に必要な機械数を計算し，全機械に同時に圃場を割り当てているため，ある機械に割り当てられた圃場数が少なくなると，別の機械で必ず多く割り当てられることになる．したがって，手法1で見られた傾向は，個体の評価として各機械の評価の和を求める時には相殺され，表3.9などに見られる様に，各機械の所要時間と一日の実際の作業時間との誤差率を表す時間に関する評価 E_{ti} は最大で0.042と小さな値となっている．また，手法2では各機械への圃場割り当てが互いに影響を及ぼし合うため，手法1の後から圃場を割り当てられる機械の移動距離が長くなるという点が改善された.

これらの二つの手法の解を比較するために，各機械に割り当てられた圃場を順に作業する時の最短巡回経路を第2章の手法により求めた．表3.14に示すように，手法2の解での各機械の移動距離の和は手法1と比較して約20%改善されている.

また，第2章の手法ではルーレット選択とランク選択を用いた計算の結果に明らかな差があり，ランク選択を用いた方が適応度の高い個体を解として得ることができた．しかし，本章で用いた手法ではこれらの選択方法による計算結果には明らかな差はなく，またMann-WhitneyのU検定により差があるとされたものではルーレット選択の方が適応度の

表 3.14: 各機械の移動距離とその和
Table 3.14: The moving distance of each machine and total moving distance

Machine No.	Method 1 (m)	Method 2 (m)
1	1296.54	172.42
2	831.04	2820.69
3	882.76	4844.82
4	1320.68	1327.58
5	1320.71	158.62
Total	11482.73	9324.13

高い個体を解として得ることができた．これは，第2章の手法と本章の手法では進化の方法が異なっているためであると考えられる．この進化の方法と選択の方法の組み合わせによるGAの性能の違いについては今後さらに検討していく必要がある．

3.6 まとめ

本章では農作業の受委託等による経営規模の拡大に伴い複雑になる作業計画を最適化する手法として，複数の機械に作業時間が均等で，圃場間の移動距離が短くなる圃場割り当てを行なう方法をGAにより構築し，次のことを明らかにした．

1. 各機械に順番に作業する圃場を割り当てる手法1では，後から圃場を割り当てられる機械ほど，圃場の位置の初期条件が悪くなり移動距離が長くなる傾向が見られた．また，最後に割り当てられる機械では，それまでに他の機械に割り当てられなかった圃場であるため，広い範囲に点在し移動距離が著しく長くなる場合が見られた．

2. 個体の適応度として時間および距離を考慮したが，手法1では割り当てられた圃場が少ない時に，時間に関する評価が悪いにも関わらず個体の適応度が高くなるものが多く見られた．このため，全圃場での作業に必要な最少台数より多くの機械が必要な結果を得ることが多く見られた．

3. これらの手法1の問題点を改善するために，あらかじめ必要最少台数を求め，全機械に同時に圃場を割り当てる手法2を開発した．その結果，各機械への圃場割り当

てが互いに影響を及ぼし合い，全機械の移動距離の和は手法1に比べて約20%改善された．

4. 手法2では，手法1と同様に各機械では割り当てられた圃場が少ない時に適応度が高くなる傾向が見られたが，ある機械で割り当てられた圃場が少ない時には，他の機械で多く割り当てられるため，個体の適応度としてはこの傾向は見られなかった．これは個体の適応度として各機械の評価の和の逆数を用いたためで，割り当てられた圃場が多い機械の評価により，割り当てられた圃場が少ない時の評価が打ち消されたためである．

5. 手法2で様々なパラメータの組み合わせで計算を行なった結果，突然変異率 P_m は7.0%もしくは10.0%に，評価関数の係数 a は1.5もしくは2.0に設定することで良好な解が得られることが判明した．

6. 本章の手法ではある機械の一日目の作業と二日目の作業が入れ替わった個体が全く同じ適応度を持つものとして扱われるため，これらの二つの解の優劣を判別することができなかった．これは評価関数に作業適期とのずれを導入することで改善できるものと思われる．また，この他に品種などの拘束条件を考慮し，より現実問題に適合させていく必要がある．

7. 第2章の手法と同様にルーレット選択とランク選択を用いて結果を比較し，本章の手法では第2章のものとは異なり，差がない，もしくはルーレット選択の方が優れているという結果を得た．これは第2章の手法と本章の手法で，進化の手順が異なっているためであると考えられる．これらの進化の手順と選択方法の組み合わせによるGAの性能への影響については今後さらに検討する必要がある．

参考文献

[1] K. Ohdoi, A. Oida and M. Yamazaki : An optimization of the working schedules in fields, Proc. of the Int. Agricultural Engineering Conference, Vol. II, pp.961–968, (1996)

[2] 大土井克明, 笈田昭, 山崎稔, 山下道弘 : 農作業計画の最適化に関する研究 (第 1 報) —作業経路の最適化—, 農業機械学会誌, 61(1), pp.91–97, (1999)

[3] 大土井克明, 笈田昭, 山崎稔, 山下道弘 : 遺伝的アルゴリズムによる作業計画の最適化 —実測値を用いた最適化—, 第 57 回農業機械学会年次大会講演要旨, pp.131–132, (1998)

[4] 川村登ら : 新版農作業機械学, 文永堂出版, (1991)

[5] 大土井克明, 笈田昭 : 農作業計画の最適化に関する研究 (第 2 報) —圃場割り当ての最適化—, 農業機械学会誌, 63(2), pp.100–108, (2001)

[6] S. Siegel 著, 藤本熙訳 : ノンパラメトリック統計学, マグロウヒル, (1985)

第4章　圃場割り当てにおける適応度の有効性

4.1　はじめに

第3章では複数の機械で作業する時に，各機械の作業時間を均等にし，圃場間の移動距離を短くすることを目的として最適化する手法について述べた．これらの目的を充たすために，GAの評価関数に時間に関する評価と距離に関する評価を導入した．距離に関する評価は，ある機械に割り当てられた圃場を順に作業する時の最短巡回経路を用いるべきであるが，第2章でも述べたように，割り当てられた圃場の数がn_fの時これらを一度ずつ訪れる経路は$(n_f-1)!/2$通り存在するためn_fが大きくなると最短巡回経路を見つけだすのは困難となる．この最短巡回経路はTSPとして様々な方法で解くことができ，本研究でも第2章で遺伝的アルゴリズムを用いて解く方法を提案したが，第3章の手法2では全圃場での作業を終了するのに必要な機械数をn_m，集団の個体数をN_p，計算終了までの世代数をN_gとすると，ある個体の適応度を計算するためにn_m回，一世代進化させるためにn_mN_p回TSPを解く必要があり，計算終了までの総計算回数は$n_mN_pN_g$回となる．作業する圃場数が増えると必要な機械数も増加し，またGAでの計算上では圃場数が増えるということは遺伝子長が長くなるということであり，それに伴って集団の個体数，最終世代までの世代交代数も増やさなければならない．したがって，作業する圃場数が増えると適応度の計算回数は爆発的に増加するため，何らかの単純化を行い計算時間を短縮する必要が生じた．本章ではこの時に用いた単純化した距離に関する評価と最短巡回経路の距離の関係を調べ，この距離に関する評価の計算方法が有効であるかを検討した．また，圃場の配置，数，面積に関わらず距離に関する評価の計算方法が有効であり，第3章の手法が適用できることを確認するために，圃場をランダムに50，および100配置した仮想の圃場データを用いて計算し結果を検討した．

4.2 計算方法

第3章の手法2では各機械についての距離に関する評価E_{di}，および時間に関する評価E_{ti}を元に各機械への圃場割り当ての評価E_iを求め，E_iの和の逆数を個体の適応度とした．これらのうち計算時間短縮のために単純化したのは距離に関する評価E_{di}であり，式(4.1)により計算した．

$$E_{di} = \frac{2}{n_{fi}(n_{fi}-1)} \sum_{j=1}^{n_{fi}-1} \sum_{k=j+1}^{n_{fi}} d_{i(j,k)} \tag{4.1}$$

ここで

E_{di}：機械iへの圃場割り当ての距離に関する評価

n_{fi}：機械iに割り当てられた圃場数

$d_{i(j,k)}$：機械iに割り当てられたj番目とk番目の圃場間の距離

で，今，機械1に割り当てられている圃場が5, 6, 23, 27, 32, 34, 35, 36, 37, 38であるとすると，$d_{1(1,2)}$は圃場5と圃場6の間の距離，$d_{1(1,3)}$は圃場5と圃場23の間の距離である．したがって，E_{di}は機械iに割り当てられた全圃場のうちの二圃場間の距離の全ての値の平均値となる．この単純化した距離に関する評価の計算方法が妥当であることを確認するために，第3章の手法2で京都府船井郡和知町の圃場データを用いた最適化中に現われた個体について，機械iの距離に関する評価E_{di}と機械iに割り当てられた圃場の最短巡回経路の距離D_i，および式(4.2)，式(4.3)により求められるE_dとDを比較した[1–3].

$$E_d = \sum_{i=1}^{n_m} E_{di} \tag{4.2}$$

$$D = \sum_{i=1}^{n_m} D_i \tag{4.3}$$

ここで

E_d：ある個体における各機械の距離に関する評価の総和

D：ある個体の表す圃場割り当てでの各機械の最短巡回経路の総和

である．

最適化中に現われた個体が示す各機械への圃場割り当てでの機械iの最短巡回経路の距離D_iは，割り当てられた圃場数n_{fi}が10以下の時は全ての巡回経路の距離を計算し，最短となるものを求めた．n_{fi}が11以上の時は第2章で用いた手法1にランク選択を組み合わせたものによりTSPとして求めた．この計算に用いたGAのパラメータを表4.1に示す．

表 4.1: パラメータ
Table 4.1: Parameters

最終世代数:N_g	400
個　体　数:N_p	200
複　製　数:N_r	$N_p - 2N_c$
交　叉　数:N_c	70
突然変異率:P_m	2.5

第 3 章の手法 2 では $t > 2T$ の時，すなわち割り当てられた圃場での所要作業時間が一日の実作業時間の 2 倍を超えた時は，その個体を環境に適応できないものとして扱った [4]. この時に用いた圃場の面積と機械の圃場作業量から計算すると，$n_{fi} \geq 28$ では必ず $t > 2T$ となり環境に適応できない個体となった．したがって GA により最短経路を求めたのは $11 \leq n_{fi} \leq 27$ となるものである．第 2 章の結果から手法 1 にランク選択を組み合わせたものにより最短経路との誤差が 1%以内の経路を得ることができることが分かっているため [5]，$11 \leq n_{fi} \leq 27$ で GA により得た距離は最短距離に十分に近いものと考えられる．

4.3 計算結果および考察

単純化した距離に関する評価が有効であるためには，割り当てられた圃場に対する単純化した距離と最短巡回経路の距離との関係が直線上に乗る必要がある．そこで，ある機械 i への圃場割り当てについて，距離に関する評価 E_{di} と最短巡回経路の距離 D_i を割り当てられた圃場数 n_{fi} ごとに最小二乗法により直線近似し相関を調べた．第 3 章の手法では，$n_{fi} < 2$ となるものを含む個体は環境に適応できない個体として扱ったので $2 \leq n_{fi} \leq 19$ であった．表 4.2 に n_{fi} が 4 から 15 の時の回帰直線の式 $D_i = \alpha E_{di} + \beta$ の係数 α，切片 β，および決定係数 R^2 を，図 4.1～図 4.12 に割り当てられた圃場数 n_{fi} が 4～15 の時の距離に関する評価と最短巡回経路の距離の関係を示す．なお $n_{fi} \geq 16$ のものについてはデータ数が少なかったため，また $n_{fi} = 2$ の時は $D_i = 2E_{di}$，$n_{fi} = 3$ の時は $D_i = 3E_{di}$ となるためここでは省略した．これは $n_{fi} = 2$ の場合は圃場間を結ぶ道路は一つだけであり E_{di} はこの距離となるのに対し，最短巡回経路はこの道路の往復であるため，また $n_{fi} = 3$ になると E_{di} は三つ存在する二圃場間の道路の距離の平均値となるのに対し，最短巡回経路はこの三つの道路の距離の和であるためである．これらの $n_{fi} = 2$，$n_{fi} = 3$，および $16 \leq n_{fi} \leq 19$ を含

めたすべてのケースでの距離に関する評価と最短巡回経路の距離の関係を図4.13に示す.

図4.1〜図4.12から割り当てられた圃場数n_{fi}が大きくなるにつれて分布の傾きが大きくなる傾向があり，表4.2の回帰直線の傾きαの値からも見ることができる．そのため，$2 \leq n_{fi} \leq 19$でのE_{di}とD_iの関係を示した図4.13では三角形状に分布している．これは，E_{di}が全圃場のうちの二圃場間の距離のすべての値の平均値を表しているのに対し，D_iは最短巡回経路の距離であるためn_{fi}個存在する二圃場間の距離の和となっているためであると考えられる．よってE_{di}に圃場数n_{fi}を掛けることで，D_iとの間に比例関係が現れる可能性があるため，D_iと$n_{fi}E_{di}$の関係を調べた．両者の回帰直線の式$D_i = \alpha' n_{fi} E_{di} + \beta'$を求め，表4.3に$n_{fi}$が4から15の時の係数$\alpha'$，切片$\beta'$を示す.

表4.2: 回帰方程式の係数と決定係数
Table 4.2: Regression coefficients and coefficient of determination

n_{fi}	α	β	R^2	n_{fi}	α	β	R^2
4	3.715	-15.67	0.9813	10	5.269	224.8	0.9376
5	4.264	-29.50	0.9723	11	5.328	313.9	0.9322
6	4.652	-18.80	0.9637	12	5.431	379.1	0.9305
7	4.887	29.56	0.9580	13	5.566	405.8	0.9229
8	5.031	98.37	0.9512	14	5.586	476.4	0.9172
9	5.165	167.7	0.9450	15	5.861	329.1	0.9325

表4.3: 回帰方程式の係数
Table 4.3: Regression coefficients

n_{fi}	α'	β'	n_{fi}	α'	β'
4	0.9287	-15.67	10	0.5269	224.8
5	0.8527	-29.50	11	0.4844	313.9
6	0.7753	-18.80	12	0.4526	379.1
7	0.6981	29.56	13	0.4281	405.8
8	0.6289	98.37	14	0.3990	476.4
9	0.5739	167.7	15	0.3908	329.1

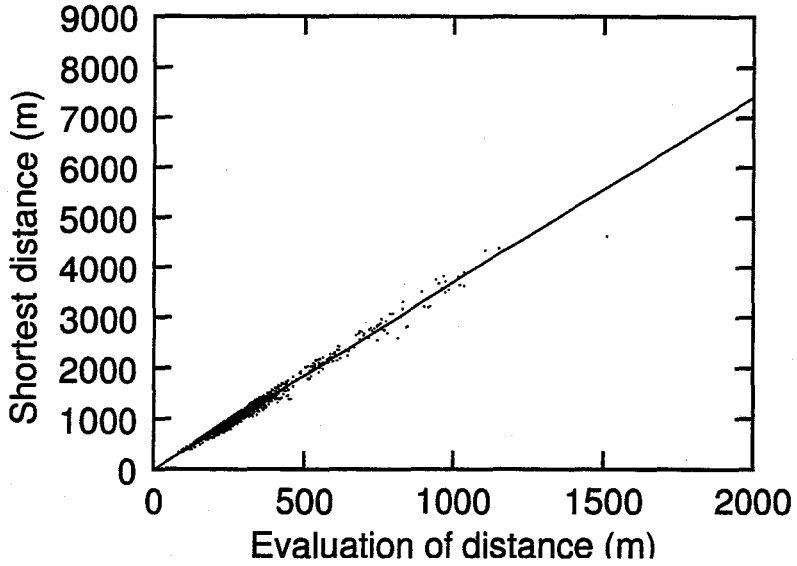


図 4.1: 距離に関する評価と最短巡回経路の距離の関係 ($n_{fi} = 4$)
Figure 4.1: The relation between evaluation of distance and shortest distance ($n_{fi} = 4$)

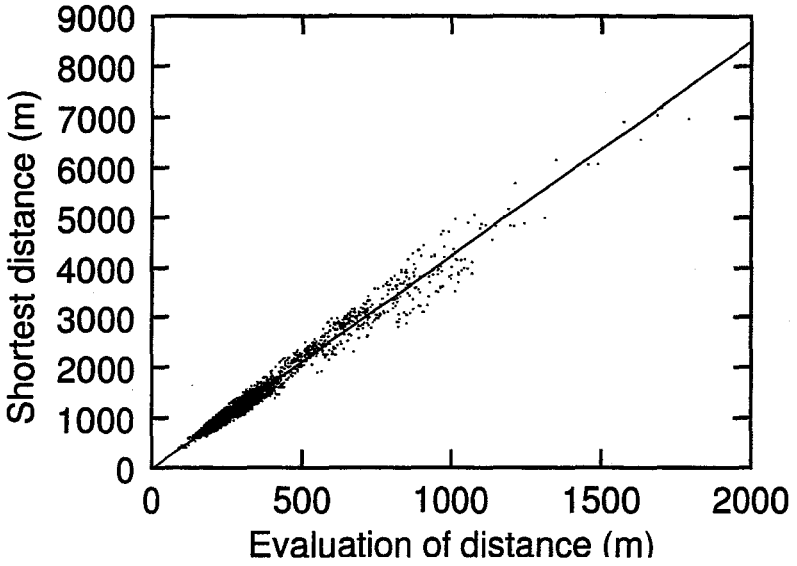


図 4.2: 距離に関する評価と最短巡回経路の距離の関係 ($n_{fi} = 5$)
Figure 4.2: The relation between evaluation of distance and shortest distance ($n_{fi} = 5$)

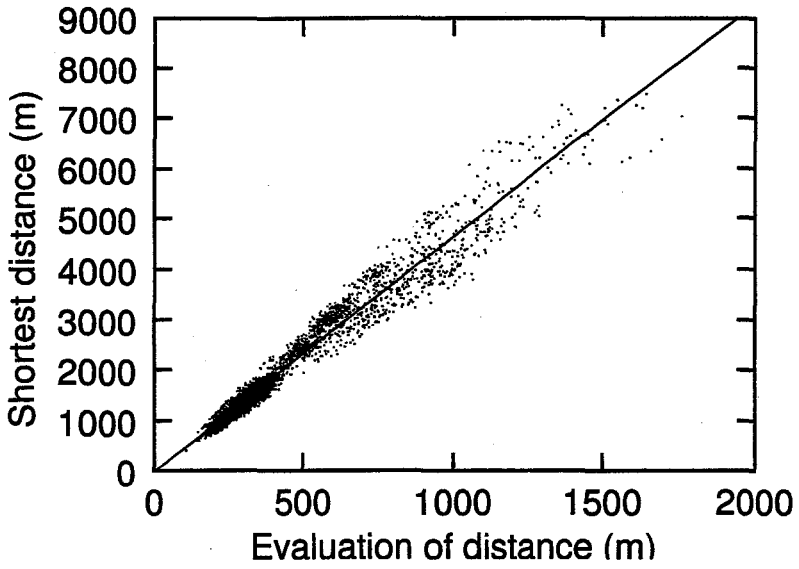


図 4.3: 距離に関する評価と最短巡回経路の距離の関係 ($n_{fi} = 6$)

Figure 4.3: The relation between evaluation of distance and shortest distance ($n_{fi} = 6$)

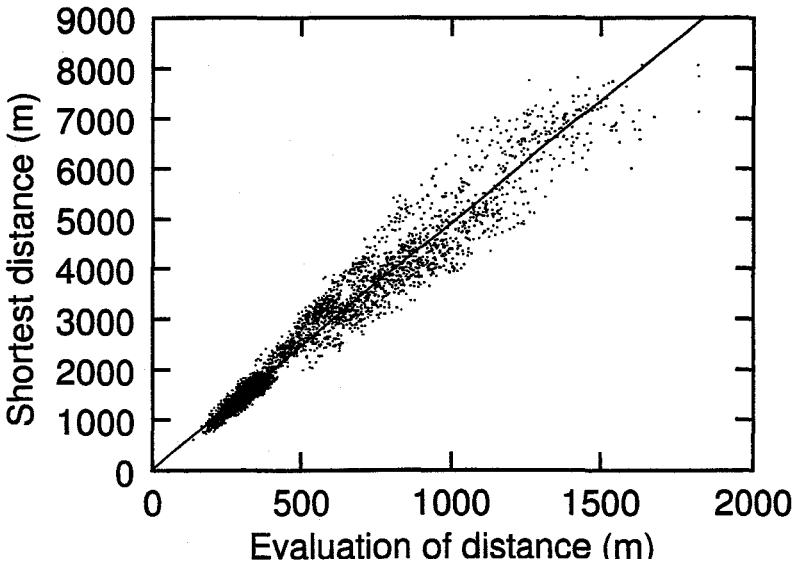


図 4.4: 距離に関する評価と最短巡回経路の距離の関係 ($n_{fi} = 7$)

Figure 4.4: The relation between evaluation of distance and shortest distance ($n_{fi} = 7$)

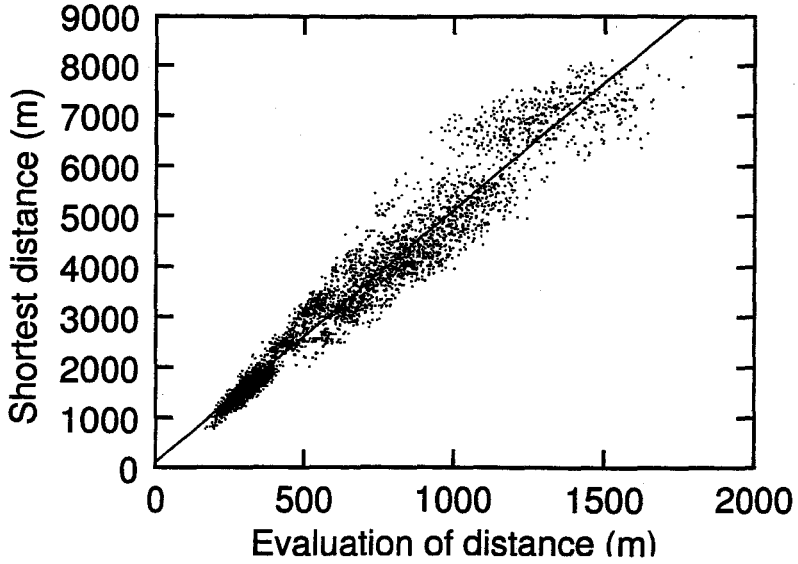


図 4.5: 距離に関する評価と最短巡回経路の距離の関係 ($n_{fi} = 8$)

Figure 4.5: The relation between evaluation of distance and shortest distance ($n_{fi} = 8$)

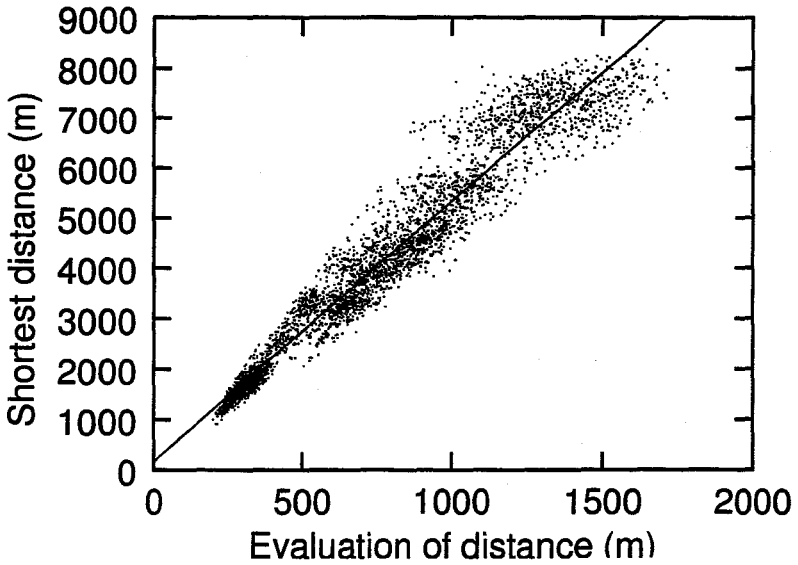


図 4.6: 距離に関する評価と最短巡回経路の距離の関係 ($n_{fi} = 9$)

Figure 4.6: The relation between evaluation of distance and shortest distance ($n_{fi} = 9$)

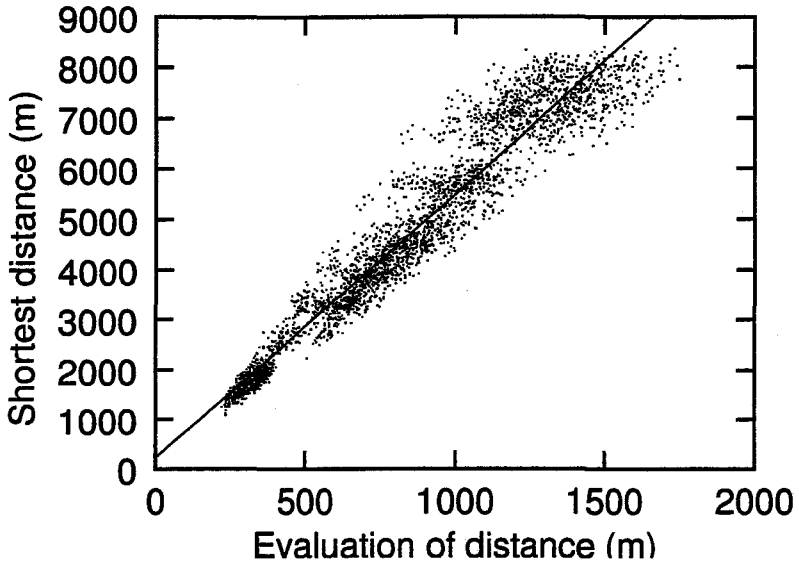


図 4.7: 距離に関する評価と最短巡回経路の距離の関係 (n_{fi} = 10)

Figure 4.7: The relation between evaluation of distance and shortest distance (n_{fi} = 10)

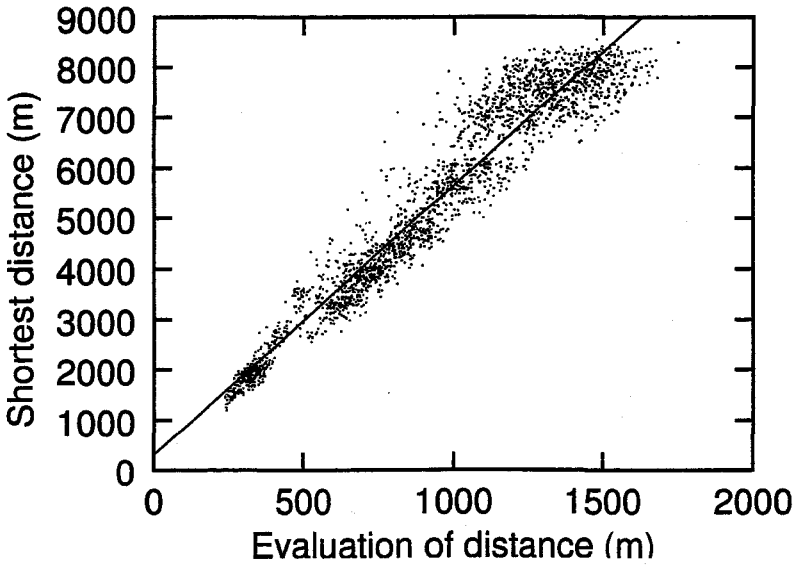


図 4.8: 距離に関する評価と最短巡回経路の距離の関係 (n_{fi} = 11)

Figure 4.8: The relation between evaluation of distance and shortest distance (n_{fi} = 11)

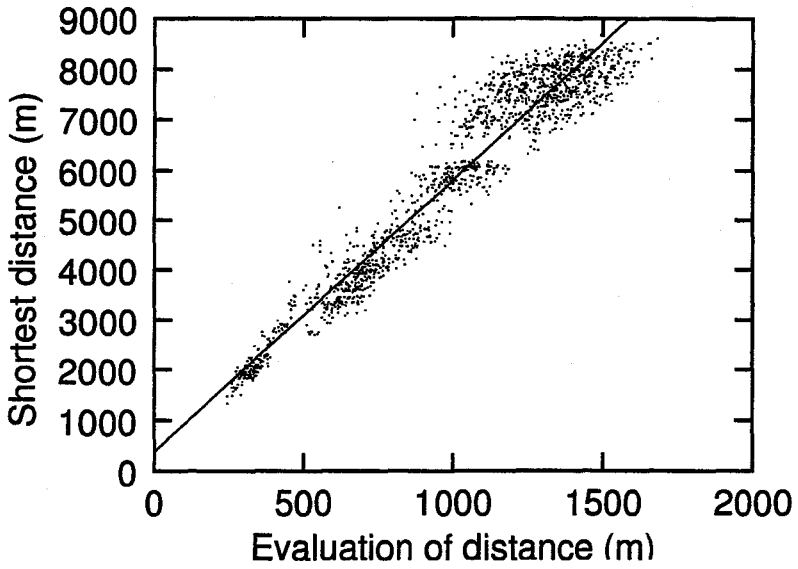


図 4.9: 距離に関する評価と最短巡回経路の距離の関係 ($n_{fi} = 12$)

Figure 4.9: The relation between evaluation of distance and shortest distance ($n_{fi} = 12$)

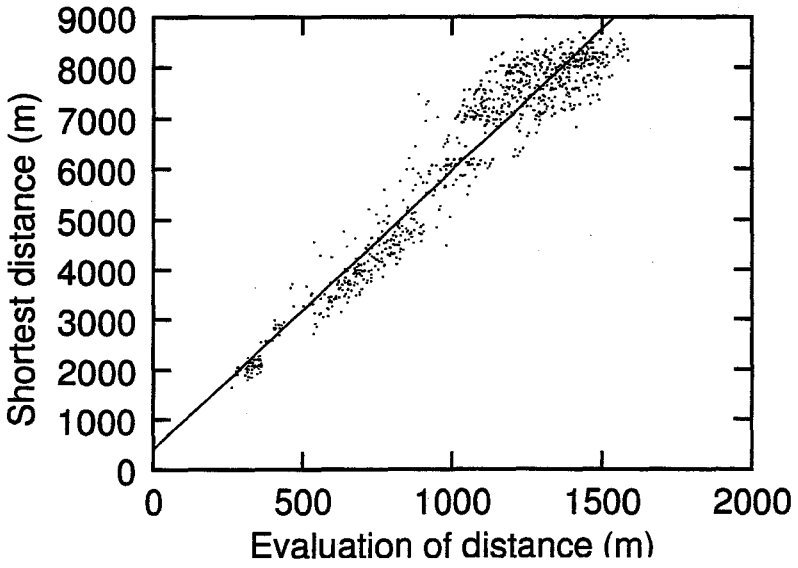


図 4.10: 距離に関する評価と最短巡回経路の距離の関係 ($n_{fi} = 13$)

Figure 4.10: The relation between evaluation of distance and shortest distance ($n_{fi} = 13$)

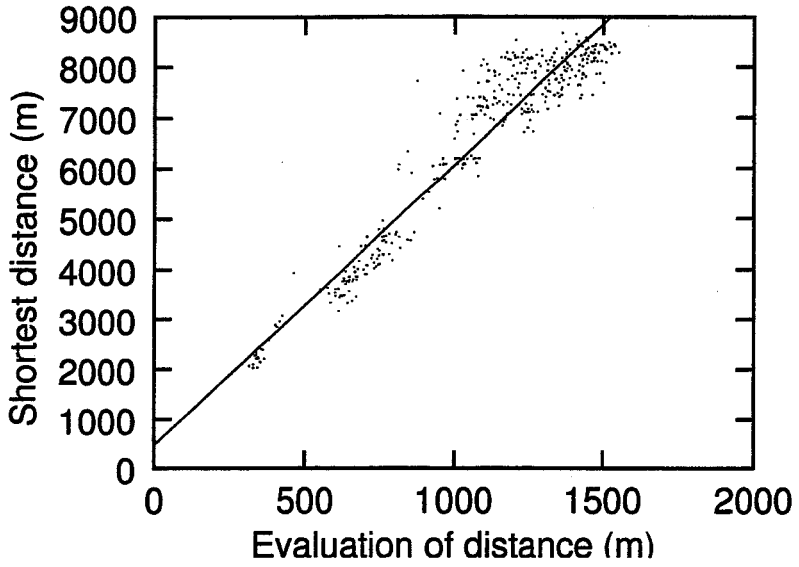


図 4.11: 距離に関する評価と最短巡回経路の距離の関係 (n_{fi} = 14)

Figure 4.11: The relation between evaluation of distance and shortest distance (n_{fi} = 14)

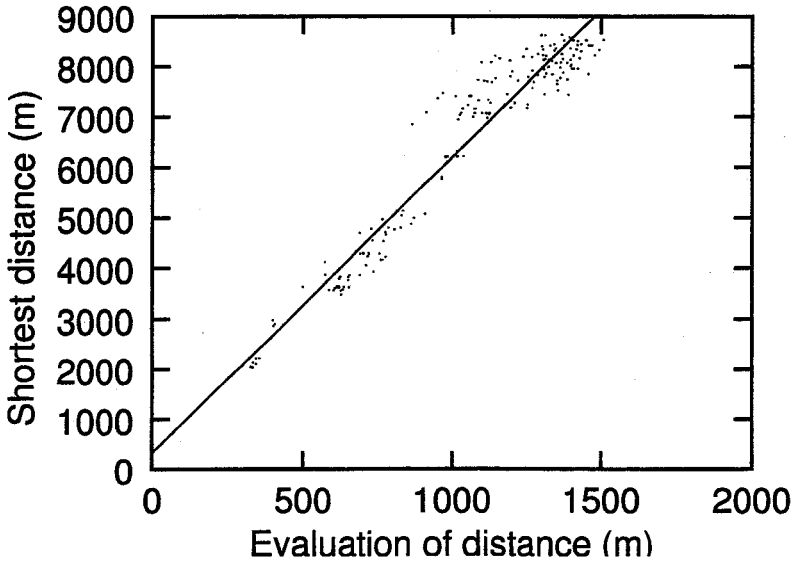


図 4.12: 距離に関する評価と最短巡回経路の距離の関係 (n_{fi} = 15)

Figure 4.12: The relation between evaluation of distance and shortest distance (n_{fi} = 15)

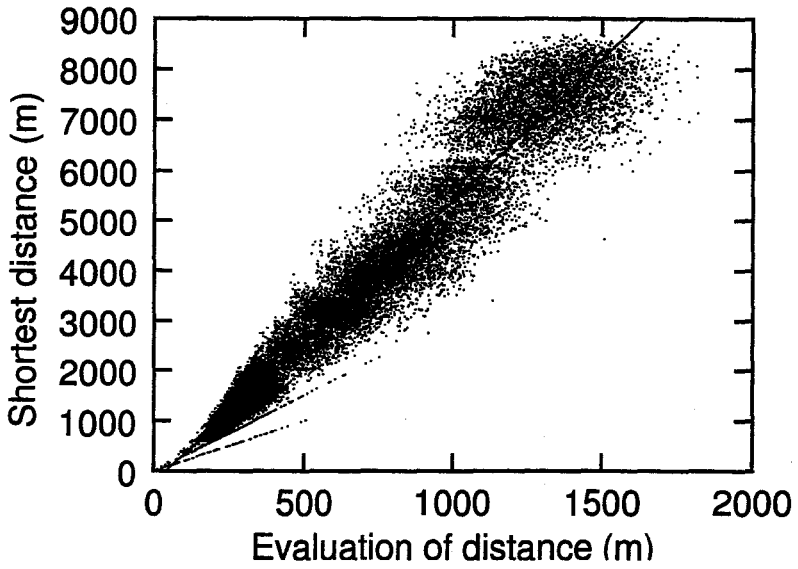


図 4.13: 距離に関する評価と最短巡回経路の距離の関係 ($2 \leq n_{fi} \leq 19$)
Figure 4.13: The relation between evaluation of distance and shortest distance ($2 \leq n_{fi} \leq 19$)

表 4.3 が示すように n_{fi} が大きくなるにしたがって直線の傾き α' は小さくなっている．これは第 3 章の考察でも述べたが，最短経路の一部とはならない経路も含めた平均値である距離に関する評価 E_{di} と n_{fi} の積は D_i より必ず大きな値となり，また最短経路の一部とはならない経路の数は $n_{fi}(n_{fi}-3)/2$ となるため，n_{fi} が大きくなるほどその影響を受けるためであると考えられる．これに対して，個体の距離に関する評価 E_d とその個体での各機械の最短巡回経路の距離の和 D の関係は図 4.14 に示すように直線性が強く $R^2 = 0.9621$ と高い相関がある．

これらのことから，全圃場での作業を終了するために必要な機械数を求め，それらの機械が作業を行なう圃場を同時に割り当てるコード化を用いた第 3 章の手法 2 では，適応度の計算方法として用いた距離に関する評価の単純化は有効であることが確認できた．また，第 3 章では手法 2 と一台ずつ順番に作業を行なう圃場を割り当てる手法 1 を比較し，手法 2 の方が使用する機械数の少ない作業計画が得られるという点で有効であると結論付けたが，図 4.13 と図 4.14 から距離に関する評価の単純化の有効性という点においても手法 2 が優れていることが証明された．すなわち，手法 1 である機械に対する圃場割り当てを行なう時，同一世代の個体群に割り当てられた圃場数が異なるものが存在するが，図 4.13 に示すように，割り当てられた圃場数が異なると距離に関する評価と最短巡回経路の

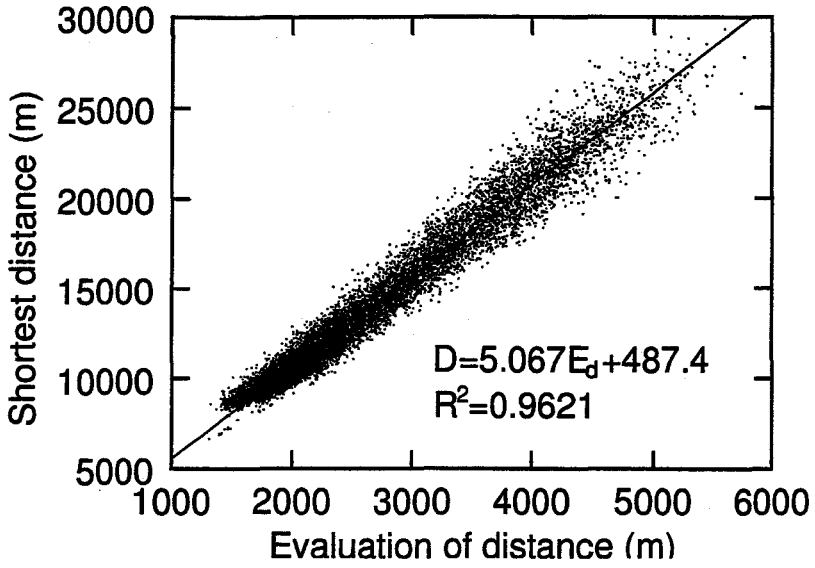


図 4.14: 個体の距離に関する評価と最短巡回経路の距離の関係
Figure 4.14: The relation between individual's evaluation of distance and shortest distance

距離の関係は三角形状に分布するためこれらの個体を正当に評価することができないことが明らかになった.

4.4 汎用性の確認

前節までの結果から，第3章で用いた手法2とその適応度の計算方法が有効であることが確認できた．そこで，この手法および適応度の計算方法が圃場の配置，数，面積に関わらず利用できることを確認するために，圃場をランダムに50，および100配置した仮想の圃場データを用いて計算した [6]．計算に用いた仮想の圃場データは，図4.15，図4.16に示すように1,000m四方の平面に面積8〜20aの圃場をランダムに50，および100配置したものである．各圃場間の距離は直線距離を，所有機械の圃場作業量，および一日の実作業時間は38圃場と同様の値を用い，38圃場の時と同様に，計算中に現れた個体の E_d と D の関係を調べた．この結果を図4.17，図4.18に示す．ともに $R^2 > 0.99$ と高い相関があり，この適応度の計算方法に汎用性があることが確認された.

図4.19，図4.20はそれぞれ全圃場数が50，および100の時の計算結果 (各機械への圃場割り当て) の一例で，使用したパラメータはそれぞれ，最終世代数 N_g : 1000, 10000，個体数 N_p : 400, 1000，交叉数 N_c : 160, 400，突然変異率 P_m : 7.0%，係数 a : 2.0である．圃

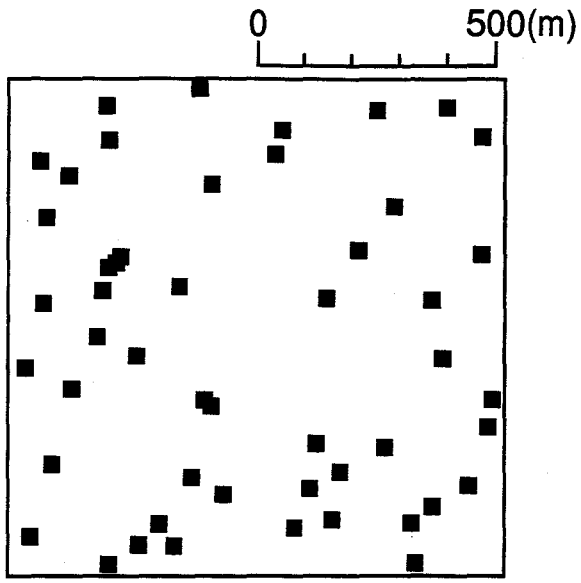


図 4.15: 圃場の配置 (全圃場数 : 50)
Figure 4.15: The distribution of fields (Total number of fields : 50)

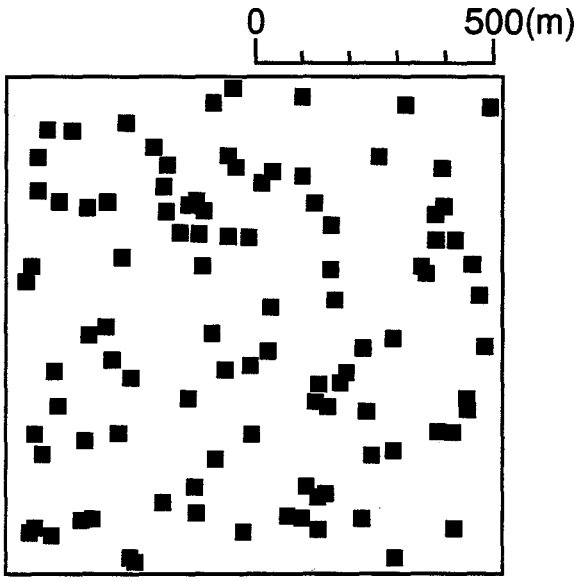


図 4.16: 圃場の配置 (全圃場数 : 100)
Figure 4.16: The distribution of fields (Total number of fields : 100)

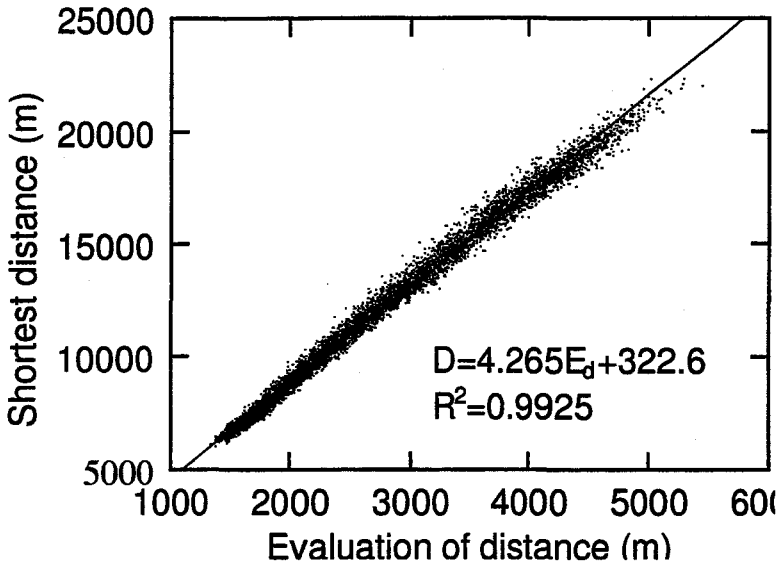


図 4.17: 個体の距離に関する評価と最短巡回経路の距離の関係

Figure 4.17: The relation between individual's evaluation of distance and shortest distance

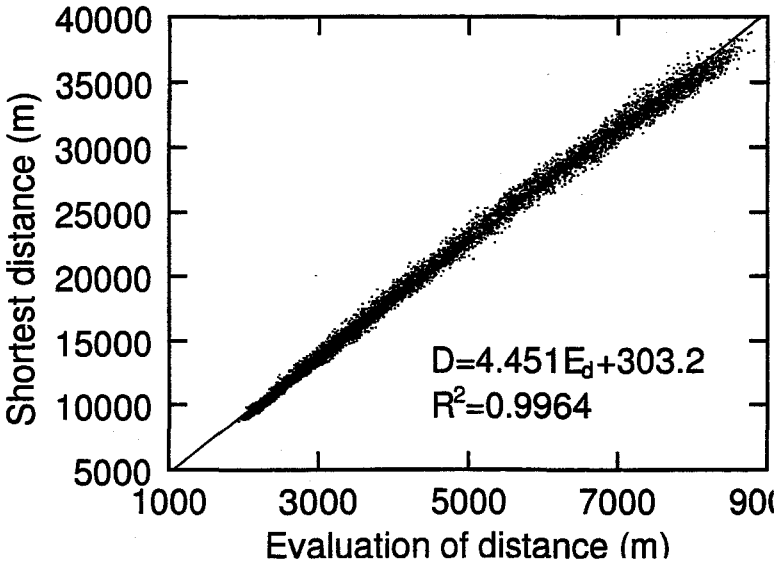


図 4.18: 個体の距離に関する評価と最短巡回経路の距離の関係

Figure 4.18: The relation between individual's evaluation of distance and shortest distance

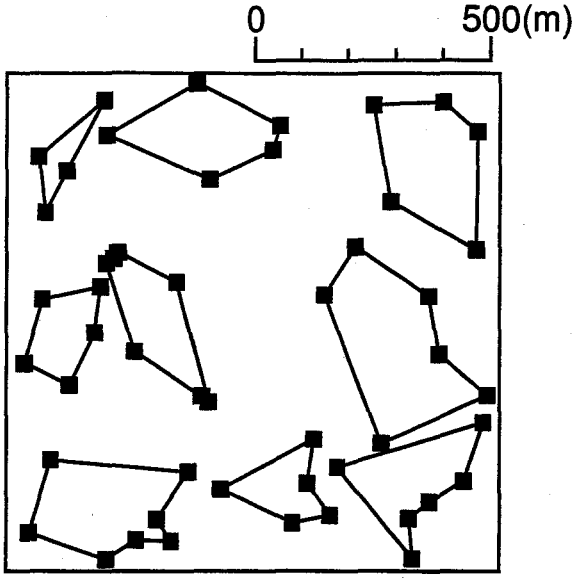


図 4.19: 各機械への圃場割り当てについての計算結果 (全圃場数 : 50)
Figure 4.19: The calculation result on field assignment for each machine (Total number of fields : 50)

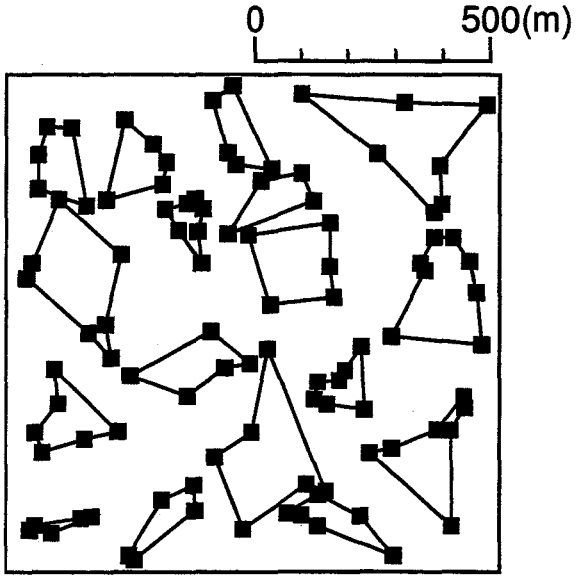


図 4.20: 各機械への圃場割り当てについての計算結果 (全圃場数 : 100)
Figure 4.20: The calculation result on field assignment for each machine (Total number of fields : 100)

表 4.4: 距離に関する評価と時間に関する評価 (全圃場数 : 50)
Table 4.4: Evaluations of distance and time (Total number of fields : 50)

Machine No.	1	2	3	4	5
E_{di}	152.03	192.72	203.61	150.35	228.17
E_{ti}	0.02	0.06	0.01	0.01	0.06
Machine No.	6	7	8	9	
E_{di}	187.88	215.35	264.21	149.30	
E_{ti}	0.03	0.01	0.02	0.01	

表 4.5: 距離に関する評価と時間に関する評価 (全圃場数 : 100)
Table 4.5: Evaluations of distance and time (Total number of fields : 100)

Machine No.	1	2	3	4	5
E_{di}	74.97	82.04	70.90	112.00	207.29
E_{ti}	0.00	0.01	0.10	0.03	0.00
Machine No.	6	7	8	9	10
E_{di}	149.50	156.12	117.81	139.16	122.06
E_{ti}	0.05	0.07	0.06	0.03	0.09
Machine No.	11	12	13	14	15
E_{di}	139.60	181.78	208.41	120.33	121.06
E_{ti}	0.03	0.04	0.01	0.10	0.03
Machine No.	16	17			
E_{di}	115.09	133.17			
E_{ti}	0.04	0.06			

場数が増えたので最終世代数，および個体数を増やしたが，個体数に対する交叉数の比率，突然変異率，および係数は第3章で良好な結果を得ることができた組み合わせをそのまま使用した．これらの計算結果で各機械に割り当てられた圃場作業の距離に関する評価 E_{di}，および時間に関する評価 E_{ti} を表4.4，表4.5に示す．これらの表から時間に関する評価 E_{ti}，すなわち割り当てられた圃場作業の所要時間と一日の実作業時間の誤差率は50圃場の場合は最大で約0.06(約18分)，100圃場の場合は最大で約0.10(約30分)と少なく，各機械に均等に圃場作業が割り当てられていることが明らかになった．また，図4.19，図4.20に示すように各機械が作業を行なう圃場は狭い範囲に存在し，それらの圃場を巡回する時の経路が他の機械の経路と交叉していないことから，圃場間の移動距離が少なくなるように圃場作業が割り当てられていると考えられる．これらのことから，第3章の手法2は圃場の配置，数，面積に関わらず適用できることが確認できた．

4.5 まとめ

本章では第3章の手法で用いた適応度の計算方法のうち，計算時間を短縮するために単純化する必要のあった距離に関する評価が個体を適切に評価していることを確認するために，第3章の手法2での最適化中に現れた個体を用いて距離に関する評価と最短巡回経路の距離の間の関係を調べた．また，この評価関数が特定の圃場配置，圃場数，面積などに依存せずに有効であり，第3章の手法に汎用性があることを確認するために，ランダムに圃場を配置した仮想の圃場データを用いて圃場割り当ての最適化を行ない，次のことを明らかにした．

1. 個々の機械への圃場割り当てについて，距離に関する評価と，その機械の最短巡回経路の距離の間の関係を調べたところ，割り当てられた圃場数が大きくなるほど両者の相関が低くなる傾向があった．また，近似直線の傾きは割り当てられた圃場数が多くなるほど大きくなり，割り当てられた圃場数が異なる時は適切な評価ができないことが判明した．

2. 各個体が示す各機械の距離に関する評価の和と，各機械の最短巡回経路の距離の和の間の関係を調べたところ，両者に高い相関が認められた．

3. 第3章の手法1では複数の機械に対して一台ずつ順番に作業を行なう圃場を割り当

てているため，最適化中に現れる個体はある機械一台に対する圃場割り当てを表す．この時，個体により割り当てた圃場数が異なっているため，1の理由から適応度の計算方法としては不適切であることが判明した．これに対して第3章の手法2では複数の機械に対して同時に作業する圃場を割り当てるため，2の理由から適応度の計算方法として適切であることが明らかになった．これらのことからも第3章で述べた，手法2が手法1より優れていることが証明された．

4. ランダムに50箇所，および100箇所の圃場を配置した仮想の圃場データを用いて，第3章の手法2で最適化を行ない，最適化中に現れた個体について各機械に割り当てられた圃場での距離に関する評価の和と各機械の最短巡回経路の距離の和を比較し，この距離に関する評価の計算方法が特定の圃場データに依存することなく有効であることを確認した．また，最適化の結果から，第3章の手法2に汎用性があることも確認した．

参考文献

[1] 大土井克明, 笈田昭, 山下道弘：農作業計画の最適化に関する研究 —評価関数の有効性の検討—, 第 59 回農業機械学会年次大会講演要旨, pp.387–388, (2000)

[2] 大土井克明, 笈田昭：農作業計画の最適化に関する研究, 農作業研究, 第 35 巻別号 1, pp.57–58, (2000)

[3] K. Ohdoi and A. Oida : A study on an optimization of farm-work schedules, Proc. of the Int. Agricultural Engineering Conference, pp.110–115, (2000)

[4] 大土井克明, 笈田昭：農作業計画の最適化に関する研究 (第 2 報) —圃場割り当ての最適化—, 農業機械学会誌, 63(2), pp.100–108, (2001)

[5] 大土井克明, 笈田昭, 山崎稔, 山下道弘：農作業計画の最適化に関する研究 (第 1 報) —作業経路の最適化—, 農業機械学会誌, 61(1), pp.91–97, (1999)

[6] 大土井克明, 笈田昭：GA による農作業計画における適応度について, 農業機械学会誌, 63(3), pp.84–89, (2001)

第5章　総括

本研究は，作業受託などにより経営規模を拡大して省力化・低コスト化を図る際に課題となる，効率的な農業機械の運用を可能とする作業計画の構築を目的としている．この作業計画の最適化を行うために，組み合わせ最適化問題に適した遺伝的アルゴリズム (GA) を応用した．GA では対象となる問題に適したコード化，遺伝的操作，評価関数を用いる必要がある．そこで，一台の機械が点在する圃場を巡回する時の最短巡回経路の探索，および複数の機械への移動距離が短く所要作業時間が均等な圃場割り当てを対象として最適化手法を構築し，それらに用いたコード化，遺伝的操作，評価関数が適切であるかを検討した．

第1章では我が国の農業における経営形態の特徴について触れ，その問題点を提起するとともに，本研究で応用したGAの概要や応用事例，対象とした作業計画に関連する研究について触れた．第2章では一台の機械が点在する圃場を巡回する時の経路の最適化について，コード化，および遺伝的操作の異なる2つの手法を最適解が既知である問題に適用して性能を比較するとともに，様々なパラメータで計算を行った結果を検討し，実際の圃場データを用いて最適化を行なった．第3章では複数の機械で作業する場合の各機械への圃場割り当てを，GAにより最適化する手法を提案した．また，評価関数を単純化し計算時間を短縮した．第4章では第3章で提案した評価関数の単純化について考察し，第3章の最適化の目的に対する有効性を確認した．

以下に各章での大要について述べる．

第2章では，一台の機械が点在する圃場を順に作業する時の最短巡回経路をGAを用いて求める手法を構築した．ある機械で複数の圃場で作業する時に，ある圃場で作業を開始するとその圃場の作業を完了するまで別の圃場に移動することはなく，また作業を終了した圃場には二度以上訪れることはないと考えることができる．したがってこの問題を，全ての都市を一度訪れ，一度訪れた都市を二度以上訪れることがないとした時の最短巡回経路を求める巡回セールスマン問題 (TSP) として考えた．GA を TSP に適用する時には致死遺伝子を持つ個体の生成が問題となるため，これを抑制する様々な手法が提案されてい

るが，本研究ではOrder Crossoverを改良した交叉を用いた手法(手法1)とコード化を順序表現とした手法(手法2)を用いた．これらの手法を最適解が既知である問題に適用し，様々なパラメータで計算した解を比較した結果，手法1が優れていることが判明した．この手法1にランク選択を組み合わせて実在の中山間地の圃場データを用いて計算を行ない，適切なパラメータを用いることで最良解と最悪解の誤差が1%以下になることが判明した．また，ニューラルネットワークを用いた手法で計算するとパラメータの決定が難しく，圃場数が変化すると同じパラメータでは対応できないという問題点があったが，本研究で用いた手法ではこれらの問題点を解消できることが判明した．農業機械の移動速度は低速であるため，運搬車に積載する時間などを考慮しても自走による移動よりも短時間で済む場合は運搬車に積載して移動するのが一般的であるが，ここで開発した手法を用いることで圃場間の移動距離を自走できる程度に短くし，結果として一日の実作業時間が増えるため作業効率向上に貢献できる．

第3章では，複数の機械で作業する時の各機械が担当する圃場を，各機械の圃場作業量を考慮して一日の実作業時間と割り当てられた圃場での所要作業時間との誤差を小さくし，圃場間の移動距離が短くなるように決定する手法をGAにより構築した．圃場間の移動距離を短くすることを目的の一つとしているため，割り当てられた圃場を巡回する時の最短巡回経路の距離を適応度に反映する必要がある．しかし，割り当てられた圃場が多くなると巡回経路の組み合わせは爆発的に増加し，最短巡回経路の距離を求めるのは困難になるため，第2章の手法などで求める必要があるが，適応度の計算は(個体数×最終世代数)回行なう必要があり，第2章の手法を用いても計算時間を考えると現実的ではない．そこで，第3章では単純化した計算方法で適応度を求めた．一台ずつ順に作業する圃場を割り当て，全ての圃場がどの機械で作業されるかが決まるまで計算を繰り返す手法(手法1)では，後から圃場を割り当てられる機械ほど，圃場の位置の初期条件が悪くなり，また割り当てられた圃場が少ない時に適応度が高くなることが多く見られた．その結果，全圃場での作業に必要な最少台数より多くの機械が必要となる結果が多く得られた．そこで，あらかじめ全圃場での作業に必要な最少台数を求め，全機械が作業する圃場を同時に決定する手法(手法2)を用いることでこれを解消した．この手法により所有する機械に所要時間が均等で狭い範囲に存在する圃場を割り当てられるため，各機械の稼働率を高めることができ，機械導入の際にシミュレーションを行なうことで農業機械への過剰投資を回避することも可能となる．

第4章では，第3章で計算時間短縮のために単純化した適応度の計算方法について考察し，その有効性を確認した．第3章の手法では割り当てられた圃場のうちの2圃場間の距離を全ての組み合わせについて求め，それらの平均値を割り当てられた圃場の最短巡回経路の距離を推測する指標として考え，距離に関する評価とした．そこで，この距離に関する評価と最短巡回経路の距離を比較した．その結果，ある機械に注目して両者を比較すると，割り当てられた圃場数が大きくなるほど相関が低くなる傾向があり，近似直線の傾きも変化するため，割り当てられた圃場数が異なる時は適切な評価ができないことが判明した．これを第3章の手法2の個体に注目し，各機械の距離に関する評価の和と最短巡回経路の距離の和を比較すると，両者に高い相関があることが認められ，第3章の手法2で用いた適応度の計算方法が有効であることが確認できた．また，ランダムに圃場を配置した仮想の圃場データを用いて計算したところ，特定の圃場データに依存することなくこの適応度の計算方法が有効であり，第3章の手法2に汎用性があることが確認できた．このことから第3章の手法2を用いてある時点での最適な圃場割り当てを行ない，一日の作業終了後にその日実際に作業した圃場を除いた圃場データで再び最適化を行なうことができる．したがって，天候や機械の故障などの理由によりある日に行なうべき作業が完了できない場合にも，次善の作業計画を得るシステムの構築が可能となる．

謝辞

本論文は，京都大学大学院農学研究科地域環境科学専攻に在学中の研究成果をとりまとめたものです．この研究を進めるにあたり多くの人からご指導やご助言を頂きました．

特に，京都大学大学院農学研究科笈田昭教授には本研究を始めるに当たって多くのヒントを頂くとともに，終始適切なご指導とご高閲を賜わりました．謹んで深謝申し上げます．京都大学大学院梅田幹雄教授，同池田善郎教授からは大変有益なご助言を賜わりました．ここに感謝の意を表します．

近畿大学山崎稔教授 (京都大学名誉教授) からは，研究室配属間もない私に研究に対する姿勢をご教示して頂くとともに，終始適切なご指導とご高閲を賜わりました．

京都大学大学院中嶋洋助教授からは研究室ゼミの場で有益なご助言を頂くとともに，コンピュータに関する多くの知識をご教示して頂きました．茨城大学清水浩助教授，京都大学大学院宮坂寿郎助手，織田敦氏をはじめとする研究室諸氏には，京都府和知町での調査を手伝って頂くとともに，様々な場での議論で有益なご助言を頂きました．

京都大学大学院村主勝彦助手には，研究を始めるに当たって本研究の遺伝的アルゴリズムの基礎となるプログラムを提供して頂き，また，プログラミングやデータ処理に関するご助言を頂きました．

京都府農林水産部山下道弘氏には京都府和知町での現地調査に際して，(財) 和知町ふるさと振興センター農作業受託部を紹介して頂き，また，多くの貴重な資料を提示して頂きました．

蒲生稔氏をはじめとする (財) 和知町ふるさと振興センター農作業受託部の皆様には，田植え，収穫といった忙しい時期での調査にも関わらず暖かく受け入れて頂きました．

以上の皆様に，謹んで感謝の意を表します．

なお，本研究は文部省科学研究費補助金 (特別研究員奨励費) の補助により行ないました．ここに記して関係各位に謝意を表します．

付録A　プログラムリスト

A.1　第2章の手法1(ランク選択)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define GENERATION 1000
#define FIELD 48
#define POPULATION 200
#define COPY (POPULATION - CROSS * 2)
#define CROSS 70
#define MUTATION 2.5
#define SUM (1 + POPULATION) * POPULATION / 2

typedef struct
{
     char Gene[FIELD];
     double Cost;
} info;

int randomx(int);
void GetDistance(void);
void MakeGene(int,int);
void Evaluation(void);
int GeneCompare(const void *,const void *);
void Limit(void);
void Alternation(void);
void Copy(void);
int Selection(void);
void Cross(void);
void Mutation(void);

info Individual[POPULATION];
info Elite;
char GeneBuf[POPULATION][FIELD];
double Distance[FIELD][FIELD];

int main(int argc, char *argv[])
{
     int i, j;
     FILE *fp0, *fp1;
```

```
    if (argc != 3) {
        printf("Usage \'method1b g_filename e_filename\'\n");
        exit(EXIT_FAILURE);
    }

    if ((fp0 = fopen(argv[1], "wt")) == NULL) {
        printf("Cannot open file %s\n", argv[1]);
        exit(EXIT_FAILURE);
    }

    if ((fp1 = fopen(argv[2], "wt")) == NULL) {
        printf("Cannot open file %s\n", argv[2]);
        exit(EXIT_FAILURE);
    }

    srand(time(NULL));
    GetDistance();
    MakeGene(0, POPULATION);
    Evaluation();
    Limit();

    for (i = 0; i < FIELD; i++)
        fprintf(fp0, "%d\t", Individual[0].Gene[i]);
    fprintf(fp0, "%f\n", Individual[0].Cost);
    fprintf(fp1, "1\t%f\n", Individual[0].Cost);

    for (i = 1; i < GENERATION; i++)
    {
        Alternation();
        Evaluation();
        if (Elite.Cost < Individual[0].Cost) {
            Individual[POPULATION-1].Cost = Elite.Cost;
            for (j = 0; j < FIELD; j++)
                Individual[POPULATION-1].Gene[j] = Elite.Gene[j];
            Evaluation();
        }

        Limit();
        for (j = 0; j < FIELD; j++)
            fprintf(fp0, "%d\t", Individual[0].Gene[j]);
        fprintf(fp0, "%f\n", Individual[0].Cost);
        fprintf(fp1, "%d\t%f\n", i + 1, Individual[0].Cost);
    }

    fclose(fp0);
    fclose(fp1);
    return 0;
}

void GetDistance(void)
{
    int i, j;
```

```
    FILE *fp;

    if ((fp = fopen("distance.dat", "rt")) == NULL) {
        printf("Cannot open distance.dat\n");
        exit(EXIT_FAILURE);
    }

    for (i = 0; i < FIELD; i++) {
        for (j = 0; j < FIELD - 1; j++)
            fscanf(fp, "%lf,", &Distance[i][j]);
        fscanf(fp, "%lf\n", &Distance[i][FIELD - 1]);
    }

    fclose(fp);
}

void MakeGene(int start, int end)
{
    int i, j, k, l;
    int GeneTable[FIELD];

    for (i = start; i < end; i++) {
        for (j = 0; j < FIELD; j++)
            GeneTable[j] = j;

        for (j = 0; j < FIELD-1; j++) {
            k = randomx(FIELD - j);
            Individual[i].Gene[j] = (char)GeneTable[k];
            for (l = k; l < FIELD - 1 - j; l++)
                GeneTable[l] = GeneTable[l + 1];
        }

        Individual[i].Gene[FIELD - 1] = (char)GeneTable[0];
    }
}

void Evaluation(void)
{
    int i, j;
    double dist;

    for (i = 0;i < POPULATION; i++) {
        dist = 0.0;
        for (j = 0; j < FIELD - 1; j++)
            dist += Distance[Individual[i].Gene[j]]
                [Individual[i].Gene[j + 1]];
        dist += Distance[Individual[i].Gene[FIELD - 1]]
            [Individual[i].Gene[0]];
        Individual[i].Cost = dist;
    }

    qsort(&Individual, POPULATION, sizeof(info), &GeneCompare);
```

```
}

int GeneCompare(const void *data1, const void *data2)
{
     const info *a, *b;
     int i;

     a = data1;
     b = data2;
     if ((a -> Cost) > (b -> Cost))
          return(1);

     else if ((a -> Cost) == (b -> Cost))
          return(0);

     else
          return(-1);
}

void Limit(void)
{
     int start, end, flag;

     start = end = flag = 0;

     while (end < POPULATION) {
          while ((Individual[end].Cost == Individual[start].Cost)
                 && (end < POPULATION))
               end++;

          if ((end - start) >= 20) {
               MakeGene(start + 1, end);
               flag++;
          }

          start = end;
     }

     if (flag != 0)
          Evaluation();
}

void Alternation(void)
{
     int i, j;

     Elite.Cost = Individual[0].Cost;
     for (i = 0; i < FIELD; i++)
          Elite.Gene[i] = Individual[0].Gene[i];

     Copy();
     Cross();
```

```
     Mutation();
     for (i = 0; i < POPULATION; i++)
          for (j = 0; j < FIELD; j++)
               Individual[i].Gene[j]=GeneBuf[i][j];
}

void Copy(void)
{
     int i, j, k;

     for (i = 0; i < COPY; i++) {
          j = Selection();
          for (k = 0; k < FIELD; k++)
               GeneBuf[i][k] = Individual[j].Gene[k];
     }
}

int Selection(void)
{
     int i;
     int r;
     int border;

     r = randomx(SUM);
     i = 0;
     border = POPULATION;
     while (border < r) {
          i++;
          border += (POPULATION - i);
     }

     return(i);
}

void Cross(void)
{
     int i, j, k;
     int dad, mom;
     int CutPos;
     int dTop, mTop;
     int dCount, mCount;
     int dClear, mClear;
     char Dad1[FIELD], Mom1[FIELD];
     char Dad2[FIELD], Mom2[FIELD];

     for (i = 0; i < CROSS; i++) {
          dad = Selection();
          mom = dad;
          while (mom == dad)
               mom = Selection();

          for (j = 0; j < FIELD; j++) {
```

```
        Dad1[j] = Individual[dad].Gene[j];
        Mom1[j] = Individual[mom].Gene[j];
    }

    CutPos=0;
    while (CutPos == 0)
        CutPos = randomx(FIELD);

    dTop = 0;
    while (Dad1[dTop] != Mom1[CutPos - 1])
        dTop++;

    mTop = 0;
    while (Mom1[mTop] != Dad1[CutPos - 1])
        mTop++;

    for (j = 0; j < FIELD; j++) {
        if (dTop == FIELD)
            dTop = 0;

        if (mTop == FIELD)
            mTop = 0;

        Dad2[j] = Dad1[dTop];
        Mom2[j] = Mom1[mTop];
        dTop++;
        mTop++;
    }

    for (j = 0; j < CutPos; j++) {
        GeneBuf[COPY + 2 * i][j] = Dad1[j];
        GeneBuf[COPY + 2 * i + 1][j] = Mom1[j];
    }

    dCount = 0;
    mCount = 0;
    for (j = 0; j < FIELD; j++) {
        dClear = 0;
        mClear = 0;
        for (k = 0; k < CutPos; k++) {
            if (Dad2[j] == Mom1[k])
                dClear++;

            if (Mom2[j] == Dad1[k])
                mClear++;
        }

        if (mClear == 0) {
            GeneBuf[COPY + 2 * i][CutPos + mCount]
                = Mom2[j];
            mCount++;
        }
```

```
                if (dClear == 0) {
                    GeneBuf[COPY + 2 * i + 1][CutPos + dCount]
                        = Dad2[j];
                    dCount++;
                }
            }
        }
}

void Mutation(void)
{
    int i, j, k;
    int buf;
    int Mut;
    int mutPos1, mutPos2;
    int mutGene;
    double fMut;
    char Mutation[FIELD];

    fMut = (double)POPULATION * (double)MUTATION / 100;
    Mut = (int)fMut;
    if ((Mut < 1) && (randomx(1000) < fMut*1000))
        Mut=1;

    for (i = 0; i < Mut; i++) {
        k = 0;
        mutGene = randomx(POPULATION - 1);
        mutPos1 = mutPos2 = randomx(FIELD - 1);
        while (mutPos1 == mutPos2)
            mutPos2 = randomx(FIELD - 1);

        if (mutPos1>mutPos2) {
            buf = mutPos1;
            mutPos1 = mutPos2;
            mutPos2 = buf;
        }

        for (j = mutPos1; j < mutPos2 + 1; j++)
            Mutation[j] = GeneBuf[mutGene][j];

        for (j = mutPos1; j < mutPos2 + 1; j++) {
            GeneBuf[mutGene][j] = Mutation[mutPos2 - k];
            k++;
        }
    }
}

int randomx(int N)
{
    return (int)((double)rand() / ((double)RAND_MAX + 1) * N);
}
```

A.2 第3章の手法2(ルーレット選択)

```
#include <stdio.h>
#include <stdlib.h>

#define PARENT(i, j) Parent[(i) * (LENGTH) + (j)]
#define DIST(i, j) DistData[(i) * (LENGTH) + (j)]
#define WRONG 1000000.0
#define FIELD_MAX 512
#define BAD 100
#define LENGTH FIELD

typedef struct
{
     char gene[FIELD_MAX];
     double time;
     double dist;
     double fitness;
} info;

void InitFunc(void);
void GetParameter(void);
void GetData(void);
void MakeChromosome(void);
void DecodeChromosome(void);
void Evolution(void);
void CrossOver(void);
void CopyGene(void);
void GeneMutate(void);
void EndFunc(void);
int CalNeedMachine(void);
int GeneCompare(const void *, const void *);
int SelectGene(void);
int randomx(int);

int FIELD;
int MACHINE;
int GENERATION;
int POPULATION;
int CROSSOVER;
int NEED;
info ELITE;
double TIME;
double R_TIME;
double MUTATION;
double MYU;
double MINIMUM;
double AVERAGE;
double MAXIMUM;
double SUM;

char *Parent;
```

```
info *Child;
double *FieldData;
double *DistData;
double *MachineData;

int main(int argc, char *argv[])
{
     int i, j, k;
     int flag = 0;
     double Result_T;
     double Result_D;
     char name[256];
     FILE *fp0, *fp1, *fp2, *fp3;

     if (argc != 2) {
          printf("usage: %s num\n", argv[0]);
          printf("num: Calculation Number\n");
          exit(EXIT_FAILURE);
     }

     sprintf(name, "f%s.dat", argv[1]);
     if ((fp0 = fopen(name, "wt")) == NULL) {
          printf("Cannot open \"%s\"\n", name);
          exit(EXIT_FAILURE);
     }

     sprintf(name, "g%s.dat", argv[1]);
     if ((fp1 = fopen(name, "wt")) == NULL) {
          printf("Cannot open \"%s\"\n", name);
          exit(EXIT_FAILURE);
     }

     sprintf(name, "r%s.dat", argv[1]);
     if ((fp2 = fopen(name, "wt")) == NULL) {
          printf("Cannot open \"%s\"\n", name);
          exit(EXIT_FAILURE);
     }

     InitFunc();
     GetData();
     NEED = CalNeedMachine();
     printf("NEED = %d, R_TIME = %f\n", NEED, R_TIME);

     ELITE.fitness = WRONG;
     MakeChromosome();
     DecodeChromosome();
     Evolution();
     fprintf(fp0, "#GENERATION\tTIME\tDISTANCE\tFITNESS\n");
     fprintf(fp0, "1\t%f\t%f\t%f\n",
             Child[0].time - TIME * (NEED - 1) - R_TIME,
             Child[0].dist, 1 / Child[0].fitness);
     for (i = 0; i < LENGTH - 1; i++)
```

```
        fprintf(fp1, "%d,", Child[0].gene[i]);
    fprintf(fp1, "%d\n", Child[0].gene[LENGTH - 1]);

    for (i = 1; i < GENERATION; i++) {
        if (((i + 1) % 100) == 0)
            printf("Generation%d\n", i + 1);
        CrossOver();
        CopyGene();
        GeneMutate();
        DecodeChromosome();
        Evolution();
        fprintf(fp0, "%d\t%f\t%f\t%f\n", i + 1,
                Child[0].time - TIME * (NEED - 1) - R_TIME,
                Child[0].dist, 1 / Child[0].fitness);
        for (j = 0; j < LENGTH - 1; j++)
            fprintf(fp1, "%d,", Child[0].gene[j]);
        fprintf(fp1, "%d\n", Child[0].gene[LENGTH - 1]);
    }

    fprintf(fp2, "Result of \"%s\"\n", argv[0]);
    fprintf(fp2, "FIELD = %d\n", FIELD);
    fprintf(fp2, "MACHINE = %d\n", MACHINE);
    fprintf(fp2, "TIME = %f\n", TIME);
    fprintf(fp2, "GENERATION = %d\n", GENERATION);
    fprintf(fp2, "POPULATION = %d\n", POPULATION);
    fprintf(fp2, "CROSSOVER = %d\n", CROSSOVER);
    fprintf(fp2, "MUTATION = %f\n", MUTATION);
    fprintf(fp2, "MYU = %f\n", MYU);

    for (i = 0; i < NEED; i++) {
        Result_T = 0.0;
        Result_D = 0.0;
        flag = 0;
        for (j = 0; j < FIELD; j++) {
            if (Child[0].gene[j] == i) {
                Result_T += FieldData[j];
                for (k = j + 1; k < FIELD; k++) {
                    if (Child[0].gene[k] == i) {
                        Result_D += DIST(j, k);
                        flag++;
                    }
                }
            }
        }

        fprintf(fp2, "%f ", Result_D / flag);
        if (i == NEED - 1)
            fprintf(fp2, "%f\n", (Result_T / MachineData[i] - R_TIME)
                    / R_TIME);
        else
            fprintf(fp2, "%f\n", (Result_T / MachineData[i] - TIME)
                    / TIME);
```

```
        }

        sprintf(name, "gene%s.dat", argv[1]);
        if ((fp3 = fopen(name, "wt")) == NULL) {
                printf("Cannot open \"%s\"\n", name);
                exit(EXIT_FAILURE);
        }

        for (i = 0; i < LENGTH - 1; i++)
                fprintf(fp3, "%d ", Child[0].gene[i]);
        fprintf(fp3, "%d\n", Child[0].gene[LENGTH - 1]);

        fclose(fp0);
        fclose(fp1);
        fclose(fp2);
        fclose(fp3);
        EndFunc();
        return 0;
}

void InitFunc(void)
{
        srand(time(NULL));
        GetParameter();

        FieldData = malloc(FIELD * sizeof(double));
        if (FieldData == NULL) {
                printf("Cannot allocate memory.\n");
                exit(EXIT_FAILURE);
        }

        DistData = malloc(FIELD * FIELD * sizeof(double));
        if (DistData == NULL) {
                printf("Cannot allocate memory.\n");
                exit(EXIT_FAILURE);
        }

        MachineData = malloc(MACHINE * sizeof(double));
        if (MachineData == NULL) {
                printf("Cannot allocate memory.\n");
                exit(EXIT_FAILURE);
        }

        Parent = malloc(POPULATION * LENGTH * sizeof(char));
        if (Parent == NULL) {
                printf("Cannot allocate memory.\n");
                exit(EXIT_FAILURE);
        }

        Child = malloc((POPULATION + CROSSOVER * 2) * sizeof(info));
        if (Child == NULL) {
                printf("Cannot allocate memory.\n");
```

```
            exit(EXIT_FAILURE);
        }
}

void GetParameter(void)
{
        FILE *fp;

        if ((fp = fopen("parameter.dat","rt")) == NULL) {
            printf("Cannot open \"parameter.dat\".\n");
            exit(EXIT_FAILURE);
        }

        fscanf(fp, "%d%d%lf%d%d%d%lf%lf", &FIELD, &MACHINE, &TIME,
                &GENERATION, &POPULATION, &CROSSOVER, &MUTATION, &MYU);
        fclose(fp);
}

void GetData(void)
{
        int i, j;
        FILE *fp0, *fp1, *fp2;

        if ((fp0 = fopen("field.dat", "rt")) == NULL) {
            printf("Cannot open \"field.dat\"\n");
            exit(EXIT_FAILURE);
        }

        if ((fp1 = fopen("distance.dat", "rt")) == NULL) {
            printf("Cannot open \"distance.dat\"\n");
            exit(EXIT_FAILURE);
        }

        if ((fp2 = fopen("machine.dat", "rt")) == NULL) {
            printf("Cannot open \"machine.dat\"\n");
            exit(EXIT_FAILURE);
        }

        for (i = 0; i < FIELD; i++) {
            fscanf(fp0, "%lf", &FieldData[i]);
            for (j = 0; j < FIELD; j++)
                fscanf(fp1, "%lf", &DIST(i, j));
        }

        for (i = 0; i < MACHINE; i++)
            fscanf(fp2, "%lf", &MachineData[i]);

        fclose(fp0);
        fclose(fp1);
        fclose(fp2);
}
```

```
int CalNeedMachine(void)
{
     int i;
     int count = 0;
     double sum = 0.0;

     for (i = 0; i < FIELD; i++)
          sum += FieldData[i];

     while (sum > 0) {
          sum -= TIME * MachineData[count];
          count++;
     }

     R_TIME = (sum + TIME * MachineData[count - 1])
          / MachineData[count - 1];
     return count;
}

void MakeChromosome(void)
{
     int i, j;

     for (i = 0; i < POPULATION + CROSSOVER * 2; i++)
          for (j = 0; j < LENGTH; j++)
               Child[i].gene[j] = randomx(NEED);
}

void DecodeChromosome(void)
{
     int i, j, k, l;
     int flag;
     double l_Time;
     double l_Dist;
     double l_Fitness;
     double sum;
     double weight;

     for (i = 0; i < POPULATION + CROSSOVER * 2; i++) {
          Child[i].time = 0.0;
          Child[i].dist = 0.0;
          Child[i].fitness = 0.0;
          for (j = 0; j < NEED; j++) {
               l_Time = 0.0;
               sum = 0.0;
               flag = 0;
               for (k = 0; k < LENGTH; k++) {
                    if (Child[i].gene[k] == j) {
                         l_Time += FieldData[k] / MachineData[j];
                         for (l = k + 1; l < LENGTH; l++) {
                              if (Child[i].gene[l] == j) {
                                   sum += DIST(k, l);
```

```
                                        flag++;
                                    }
                                }
                            }
                        }

                        if (flag == 0)
                            l_Dist = WRONG;
                        else
                            l_Dist = sum / flag;

                        if (j == NEED - 1) {
                            if ((fabs(l_Time - R_TIME) / R_TIME) <= 1)
                                weight = MYU * fabs(l_Time - R_TIME)
                                    / R_TIME + 1;
                            else
                                weight = BAD;
                        }
                        else {
                            if ((fabs(l_Time - TIME) / TIME) <= 1)
                                weight = MYU * fabs(l_Time - TIME)
                                    / TIME + 1;
                            else
                                weight = BAD;
                        }

                        l_Fitness = l_Dist * weight;
                        Child[i].time += l_Time;
                        Child[i].dist += l_Dist;
                        Child[i].fitness += l_Fitness;
                    }
                }
            }

            void Evolution(void)
            {
                int i, j;
                double sum = 0.0, ave = 0.0;

                qsort(Child, POPULATION + CROSSOVER * 2, sizeof(info),
                      &GeneCompare);

                if (ELITE.fitness < Child[0].fitness) {
                    Child[POPULATION + CROSSOVER * 2 - 1] = ELITE;
                    qsort(Child, POPULATION + CROSSOVER * 2, sizeof(info),
                          &GeneCompare);
                }

                else
                    ELITE = Child[0];

                for (i = 0; i < POPULATION; i++) {
```

```
            for (j = 0; j < LENGTH; j++)
                PARENT(i, j) = Child[i].gene[j];
            ave += Child[i].fitness;
            sum += (1 / Child[i].fitness);
        }

        MINIMUM = Child[0].fitness;
        AVERAGE = ave / POPULATION;
        MAXIMUM = Child[POPULATION - 1].fitness;
        SUM = sum;
}

int GeneCompare(const void *data1, const void *data2)
{
        const info *p1 = data1;
        const info *p2 = data2;

        if ((p1 -> fitness) > (p2 -> fitness))
            return 1;
        else if ((p1 -> fitness) == (p2 -> fitness))
            return 0;
        else
            return -1;
}

void CrossOver(void)
{
        int i, j;
        int dad, mom, cutpos;

        for (i = 0; i < CROSSOVER; i++) {
            dad = SelectGene();
            mom = dad;

            while (mom == dad)
                mom = SelectGene();

            cutpos = 0;

            while (cutpos == 0)
                cutpos = randomx(LENGTH);

            for (j = 0; j < cutpos; j++) {
                Child[POPULATION + i * 2].gene[j] = PARENT(dad, j);
                Child[POPULATION + i * 2 + 1].gene[j] = PARENT(mom, j);
            }

            for (j = cutpos; j < LENGTH; j++) {
                Child[POPULATION + i * 2].gene[j] = PARENT(mom, j);
                Child[POPULATION + i * 2 + 1].gene[j] = PARENT(dad, j);
            }
        }
```

```
}

int SelectGene(void)
{
     int i = 0;
     double r, border = 0.0;

     r = SUM * ((double)randomx(10000) / 10000.0);
     while (border <= r) {
          border += (1 / Child[i].fitness);
          i++;
     }

     if (i == 0) {
          printf("error1\n");
          exit(-1);
     }

     return (i - 1);
}

void CopyGene(void)
{
     int i, j;

     for (i = 0; i < POPULATION; i++) {
          for (j = 0; j < LENGTH; j++)
               Child[i].gene[j] = PARENT(i, j);
     }
}

void GeneMutate(void)
{
     int i;
     int MutateGene;
     int MutatePos;
     int N_Mutate = (int)(POPULATION * LENGTH * MUTATION / 100);

     for (i = 0; i < N_Mutate; i++) {
          MutateGene = randomx(POPULATION);
          MutatePos = randomx(LENGTH);
          Child[MutateGene].gene[MutatePos] = randomx(NEED);
     }
}

void EndFunc(void)
{
     free(FieldData);
     free(DistData);
     free(MachineData);
     free(Parent);
     free(Child);
```

```
}

int randomx(int N)
{
     return (int)((double)rand() / ((double)RAND_MAX + 1) * N);
}
```